DOCUMENT RESUME

ED 352 991 IR 054 343

AUTHOR Kriz, Harry M.
TITLE A Public-Use, Full-Screen Interface for SPIRES Databases.
PUB DATE Oct 92
NOTE 57p.; This paper was prepared for distribution at the SPIRES Fall Workshop (Chapel Hill, NC, October 12-14, 1992).
PUB TYPE Guides - Non-Classroom Use (055) -- Computer Programs (101) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS Access to Information; College Libraries; *Computer System Design; *Database Management Systems; Databases; *Gateway Systems; Higher Education; Information Retrieval; Online Catalogs; *Online Searching
IDENTIFIERS Newspaper Index (Database); Virginia Polytechnic Inst and State Univ

ABSTRACT

This paper describes the techniques for implementing a full-screen, custom SPIRES interface for a public-use library database. The database-independent protocol that controls the system is described in detail. Source code for an entire working application using this interface is included. The protocol, with less than 170 lines of procedural code, can be used virtually without modification with any SPIRES subfile for which appropriate full-screen formats have been designed. The interface enables end users with no knowledge of SPIRES, and with no training on the system, to conduct complex Boolean searching across data fields. The user does not need to issue any commands. Rather, all necessary actions are initiated by pressing a function key after entering keywords on a full-screen search form. A help system presents necessary explanations and enables easy browsing of indexed keywords and authority terms. The system is illustrated with examples from Newspaper-Index, a public use database now in use in the libraries at Virginia Polytechnic Institute and State University. The appendices to this paper contain the complete source code for the Newspaper Index application as it was working in the Virginia Tech Libraries on 9/24/92 using SPIRES version 89.03 under IBM's CMS operating system. (Author/KRN)

A Public-Use, Full-Screen Interface for SPIRES Databases

Harry M. Kriz
Automation Librarian
University Libraries
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
September 25, 1992

Prepared for distribution at the SPIRES Fall Workshop
University of North Carolina at Chapel Hill
October 12-14, 1992

"...we are never looking for the best program, seldom looking for a good one, but always looking for one that meets the requirements."

Gerald M. Weinberg
The Psychology of Computer Programming
Van Nostrand Reinhold, N. Y., 1971, p. 17

TABLE OF CONTENTS

ABSTRACT

This paper describes the techniques for implementing a full-screen, custom SPIRES interface for a public-use library database. The database-independent protocol that controls the system is described in detail. Source code for an entire working application using this interface is included. The protocol, with less than 170 lines of procedural code, can be used virtually without modification with any SPIRES subfile for which appropriate full-screen formats have been designed. The interface enables end users with no knowledge of SPIRES, and with no training on the system, to conduct complex Boolean searching across data fields. The user does not need to issue any commands. Rather, all necessary actions are initiated by pressing a function key after entering keywords on a full-screen search form. A help system presents necessary explanations and enables easy browsing of indexed keywords and authority terms. The system is illustrated with examples from NEWSPAPER INDEX, a public-use database now in use in the libraries at Virginia Polytechnic Institute and State University.

INTRODUCTION

During the past several years, a number of SPIRES databases were implemented in the libraries at Virginia Polytechnic Institute & State University (Virginia Tech). Most of these were devised by public services librarians to manage information about, and provide enhanced access to, specialized areas of the collection. As these databases grew in size and scope, it became apparent that making the systems available for direct use by the public was desirable.

Direct public use would relieve librarians from an increasingly heavy load of custom searches requested by library users. Users would be able to get results quickly at their own convenience. Such public, end user access would extend the reach of the databases to a broader range of individuals both on campus and throughout the state of Virginia.

In making the databases available to the public, it was recognized that only a handful of students, faculty, and librarians at Virginia Tech have any knowledge of SPIRES. To avoid the need for implementing a major training program, it was necessary that the public user interface not require any knowledge of SPIRES commands. The naive user should be able to access the system and begin searching immediately. This implied the user should be able to fill out an on-screen form describing the desired result. Function keys should provide all user functions, including conducting the search, viewing the results, printing the results, and asking for help. SPIRES contains all the tools necessary to create such a system with a minimum amount of programming. Borrowing a term popular in the microcomputer world, it can be said that creation of such an interface is within the reach of the SPIRES "power user."

Creating a full-screen SPIRES interface for the first time can be confusing. Knowledge of several parts of SPIRES is necessary, and real testing cannot begin until all the parts work together well enough so that some user input can be read from the screen and some output data can be placed on the screen. The document *SPIRES Device Services* (May 12, 1989, CMS edition) devotes a chapter to implementing a full-screen interface. As with all SPIRES documentation, this manual is clear, concise, and accessible to the SPIRES "power user." Additional documentation is distributed through the manuals *SPIRES Protocols* and *SPIRES Formats*. Fortunately, the bulk of the code needed to create the interface is descriptive, and not procedural. It is this feature of SPIRES that makes it so usable by non-programmers.

Considering the distributed nature of the documentation, it was felt that a more complete description of a working library database in use by the public would be helpful to those attempting to write their first full-screen application. This document presents such a description, including complete source code for Virginia Tech's NEWSPAPER INDEX. Included here is a full-screen protocol that can be used almost without modification to construct a full-screen interface for any SPIRES subfile. The protocol contains less than 170 lines of procedural code, which is almost all the procedural code necessary to implement a basic full-screen interface. Unlike the

sample protocol in the 1989 documentation, the protocol presented here takes advantage of the structured programming facilities that have been added to the SPIRES protocol language in recent years. By using function keys to initiate all actions, it also eliminates the need for the user to type any commands. Those attempting to write their own interface can use this protocol as printed simply by observing certain naming conventions when designing screen formats.

It is not necessary that the reader be experienced with SPIRES formats and protocols to follow this discussion. Concepts will be described in general terms so as to lead the reader to focus on the most relevant sections of the SPIRES documentation. Readers who are comfortable creating custom applications with microcomputer database software may be surprised at the straightforward means by which a user interface can be implemented on a mainframe using SPIRES. Such microcomputer power users can easily become power users of SPIRES thanks to the capabilities of the non-procedural, descriptive SPIRES languages.

The discussion that follows is based on the documentation for the 1989 release of the CMS version of SPIRES. The source code in the appendices is that which was in use by NEWSPAPER INDEX in the Virginia Tech Libraries on 9/24/92.

NEWSPAPER INDEX--A SAMPLE SYSTEM

The methods for creating a full-screen interface for a SPIRES database will be illustrated by reference to the Virginia Tech Library's NEWSPAPER INDEX. This application was chosen from among several in existence at Virginia Tech because the simplicity of the database structure allows the reader to focus on the details of the interface.

NEWSPAPER INDEX is produced by reference librarians who scan two area newspapers plus two campus newspapers for articles of interest to the Virginia Tech community. The area newspapers of interest are the *Christiansburg News Messenger* (the county newspaper) and the *Roanoke Times & World News* (published in nearby Roanoke, a city with a population of about 100,000). Campus newspapers indexed in the system are the *Collegiate Times* (the student paper) and the *Virginia Tech Spectrum* (the weekly newspaper for faculty and staff). Articles relating to Virginia Tech and the southwest Virginia locale are included. The database was implemented originally on index cards, transferred to a microcomputer in the mid-1980's, and converted to SPIRES during Summer, 1991. At that time, some 30,000 articles had been indexed. The system has grown since to contain information about nearly 45,000 newspaper articles.

SAMPLE SESSION

NEWSPAPER INDEX can most easily be understood by examining the following sequence of screens showing an illustrative (but certainly atypical) online session. In this sample session, the user is searching NEWSPAPER INDEX for newspaper articles about the popular topic of beer and alcohol in relation to students. After logging on, the user enters SPIRES and selects the subfile NEWSPAPER INDEX. At Virginia Tech this can be done by selecting NEWSPAPER INDEX from a menu on the mainframe INFO system so the user never needs to know anything about the SPIRES system. The figure captions trace the user's steps, and they indicate what components of the system are executing when each screen is displayed.

Fig 1. Upon selecting, Newspaper Index, the user sees this identifying screen. (Frame LOGO in format MSG.)

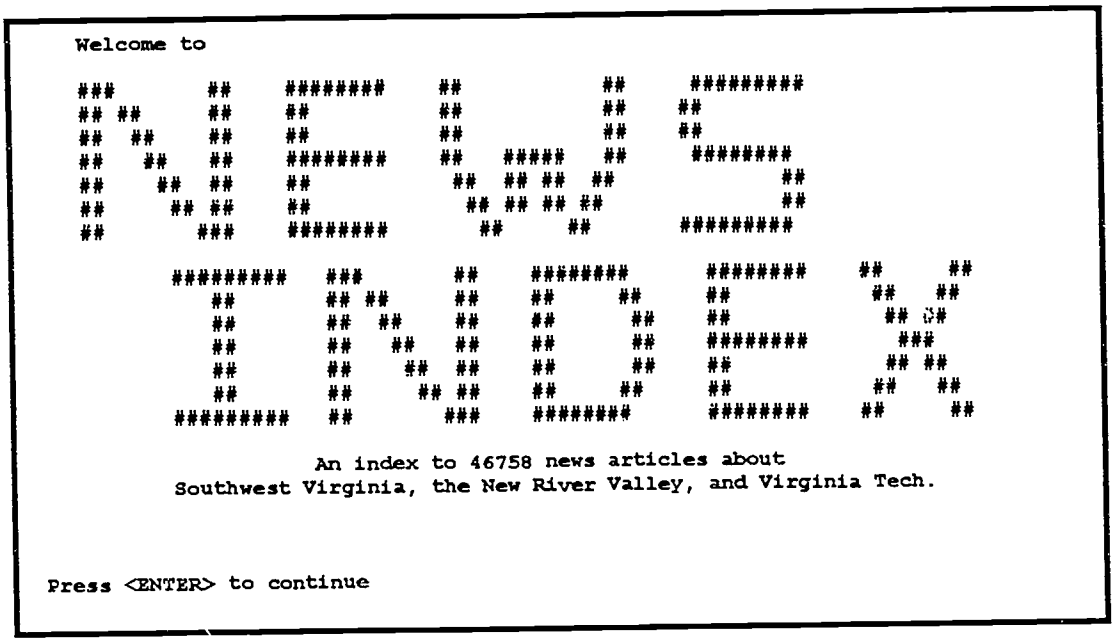

Fig 2. A brief explanatory message describes the database and provides some initial guidance in its use (Frame GREETING in format MSG.)

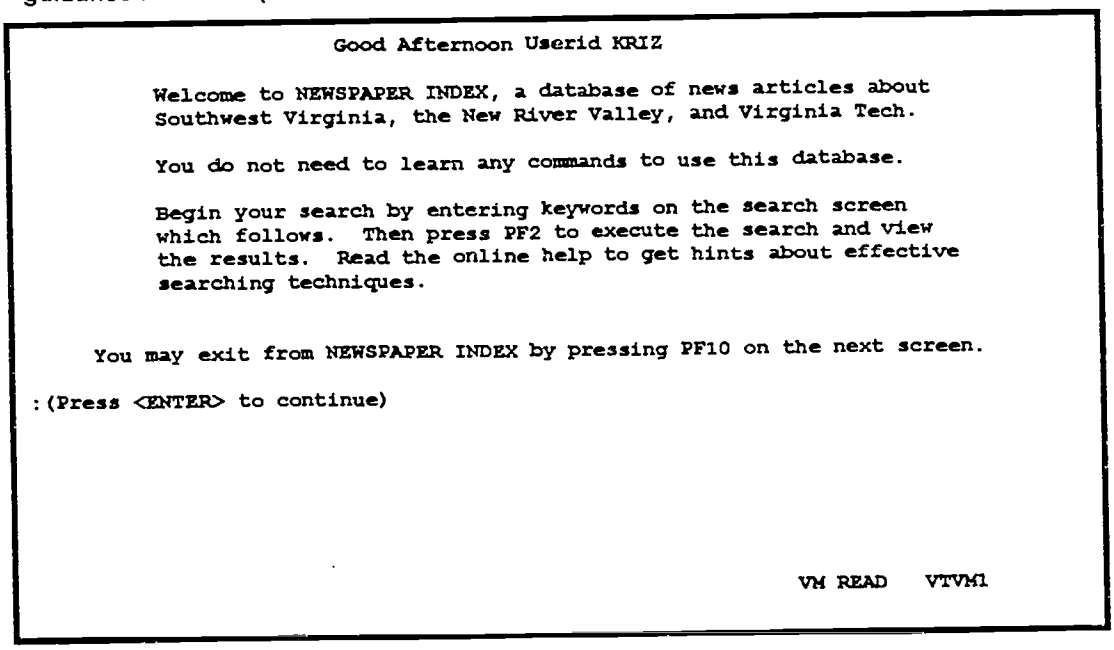

Fig 3. A blank search screen is presented to the user, making it obvious that keywords and dates may be entered for searching. The adventurous new user enters a compound search request (shown in italics) using Boolean operators. The user presses PF2 to conduct the search. (Blank data entry screen created by frame SRCHFORMDISP in format PUBLIC, data entered by user read from screen by frame SRCHFORMGET.)

```
                              NEWSPAPER SEARCH
Enter keywords, limit dates, then press PF2

PF1=Help  PF2=Do search  PF3=Erase words  PF4=Show again          PF10=Exit
═══════════════════════════════════════════════════════════════════════════

Enter subject and title keywords on the lines below:
==>  (beer or alcohol) student
==>
==>

EXCLUDE articles published before (mm/dd/yy):

 EXCLUDE articles published after (mm/dd/yy):
```

Fig 4. The syntax entered by the inexperienced user is not understood by the system, which displays an informative message. (Output typed to screen by ERRORS protocol.)

```
*
*  Your keyword syntax is interpreted as
*
*     KW (beer or alcohol) student
*
*  which is a form NOT understood by the system.
*  Please edit your keywords and try again.
*
*  When you press the CLEAR key, you will be returned
*  to the search screen and you will see the message
*
*        'No articles found...'
*
*  You may continue searching after modifying your request.




                                                    MORE...    VTVM2
```

Fig 5. On returning to the search screen, the user presses PF1 to get help and sees this menu. He enters the number ***90*** to indicate he wants a list of keywords. He enters the string ***alcoho*** as a stem for keywords. (Menu displayed by frame HELP.S.MENU in format HELP, user entry read by frame HELP.GET in format HELP.)

```
                              HELP FOR SEARCHING
Enter the number of the desired topic, press PF2

         PF2=View selection        PF3=Quit HELP
═══════════════════════════════════════════════════════════════════

  Enter number of topic ==> 90

     1) General information          7) Boolean operators

     2) Function keys                8) Compound search expression

     3) Keywords                     9) Natural language searching

     4) Dates                       10) Wild card searching

     5) Personal names              11) Costs

     6) Newspapers                  12) Technical details

                                    13) Special services


    90) List keywords beginning with: alcoho
```

Fig 6. A list of keywords is shown to the user, indicating there are several terms of interest beginning with the string ***alcoho***. (Displayed by the BROWSE index command issued from the HELP protocol.)

```
ALCATEL
ALCC
ALCHOLIC
ALCO
ALCOA
ALCOHOL
ALCOHOLIC
ALCOHOLICS
ALCOHOLISM
ALCOHOLS
-More?
y
ALCORN
ALCOTTS
ALDEN
ALDER
ALDERMAN
ALDERSON
ALDICARB
ALDRIDGE
ALE
ALEASE
                                                    MORE...   VTVM1
```

Fig 7. The user also looks up help for wildcard searching. (Displayed by frame SHOWHELP in format HELP.)

```
                        WILDCARD SEARCHING

When searching a natural language index, you must consider the possible
variations in which a word may appear.  Words may be singular, plural,
and possessive.  A word of interest may also serve as the stem of
several other words which are of interest.

For example, when searching for articles on student abuse of alcohol,
you might search for each of the keywords ALCOHOL, ALCOHOLIC,
ALCOHOLICS, ALCOHOLISM, and possessive forms such as ALCOHOLIC'S.  You
could enter each of these words separated by an OR operator to create a
single search expression that would retrieve all references as part of
a single search result.  (See Boolean Operators).

The same result can be obtained more easily by entering the keyword
ALCOHOL*, where the * is a 'wild card' which stands for any string of
characters following the word stem ALCOHOL.

When using the wild card character, make the word stem at least 5
characters long.  The system requires at least 5 characters so as to
reduce the number of irrelevant articles.  To take an extreme example,
it would serve little purpose to search on the keyword A* because that

                                                    MORE...   VTVM1
```

Fig 8. The user enters a properly formatted compound search request (shown in italics) and presses PF2 to conduct a search. (Blank data entry screen created by frame SRCHFORMDISP in format PUBLIC, data entered by user read from screen by frame SRCHFORMGET.)

```
                          NEWSPAPER SEARCH
Enter keywords, limit dates, then press PF2
No articles found for KW (beer or alcohol) student
PF1=Help  PF2=Do search  PF3=Erase words  PF4=Show again          PF10=Exit
═══════════════════════════════════════════════════════════════════════════

Enter subject and title keywords on the lines below:
==>  student* and (beer or alcohol*)
==>
==>

EXCLUDE articles published before (mm/dd/yy):

 EXCLUDE articles published after (mm/dd/yy):
```

Fig 9. The user displays one of the records in his search result. (Displayed by frame SRCHRESULT in format PUBLIC.)

```
                                    ARTICLES
FIND: KW student* and (beer or alcohol*)
Record 10 of 18
PF1=Help  PF3=Quit viewing  PF7=Previous article  PF8=Next article  PF9=Print
═══════════════════════════════════════════════════════════════════════════

Source: Roanoke Times      Date: 09/13/87   Page: NRV 1
 Title: Researchers ready to party: Student's return means it's time to study
        drinking
 Subjects: ALCOHOL.  GELLER.  PSYCHOLOGY.  RUSS.  SERVER.  TESTING.  TIPS.
           TRAINING FOR INTERVENTION PROCEDURES BY SERVERS OF ALCOHOL.
           VIRGINIA TECH.
```

Fig 10. Upon pressing PF9, the user has the opportunity to print the results of the search.

```
*
*  There are 18 titles to be printed
*  for the search
*
*     KW student* and (beer or alcohol*)
*
*   Printout will be generated on the 3800 printer
*   in the Computing Center.
*
:Do you wish to print the list of articles (Y/N)?
```

VM READ VTVM2

FEATURES AND LIMITATIONS

Features of this full-screen interface include the following:

- User needs no knowledge of SPIRES.
- User needs no training because commands are function-key driven and selected from a menu.
- A menu-driven help system can be invoked by pressing PF1.
- While entering a new search, the results of the most recent successful search can be redisplayed starting at the record last viewed. This allows referral to an existing result while entering a new set of keywords for a subsequent search.
- Results of the most recent successful search can be viewed even following execution of a search that does not retrieve any records.
- Display of records wraps around so that the first record in a result is displayed after the last record, and the last record is displayed before the first record.
- User entries on search screens and help screens are remembered by the system so the user can revise a search simply by editing the existing keywords without having to retype an entire request.
- The number of records allowed for display in a result is controlled by the designer.
- An error system provides clearly worded explanations and instructions, isolating the user from cryptic system messages.

There are two limitations to this full-screen interface:

- Only one record is displayed on the result screen.
- All data elements to be displayed must fit on one screen.

COMPONENTS OF A FULL-SCREEN INTERFACE

The sample session with NEWSPAPER INDEX indicates that a full-screen interface is conceptually simple. First, there is a search screen on which the user enters keywords and other parameters, such as dates, which define the desired information. The user does not need to know anything about the structure of the data elements and indexes, nor does he need any knowledge of SPIRES commands. In NEWSPAPER INDEX, the search screen reveals to the user that a search can be made for articles described by keywords, that the search can be limited by the date an article was published, and that certain actions will occur when a function key is pressed. Second, there is a result screen that displays records from the database that satisfy the search criteria.

A complete system adds screen displays which welcome the user when the subfile is first selected and, and which provide clear explanations and instructions when SPIRES cannot execute the request entered by the user. A help system can guide the user in formulating a search request so as to take advantage of the capabilities built into the SPIRES subfile, such as wild card searching. It also can provide information about the contents of the database, including dates of coverage and the treatment of personal names.

Implementing this conceptually simple system requires knowledge of several SPIRES subsystems. The heart of the application is a program named PUBLIC written in the SPIRES PROTOCOL language. The PUBLIC protocol as used in NEWSPAPER INDEX uses less than 170 lines of procedural code to implement searching and display of records. Once the basic system for search and display is working, three additional protocols can be added as modules to implement the help and printing functions, and to display custom error messages.

Much of the code in the protocols consists of SPIRES commands that a user could issue from the keyboard. The PUBLIC protocol issues commands that use SPIRES formats to format the screen displays, and to read the keywords entered by the user on the search screen. A SPIRES format consists mostly of non-procedural code that describes the content and location of data to be written to or read from a physical device such as a computer screen. Both the protocol and the format use variables to exchange information between themselves and the SPIRES system. Variables are defined in SPIRES vgroups. Learning how protocols, formats and vgroups work together is the major hurdle to be overcome in writing one's first full-screen interface.

The complete NEWSPAPER INDEX application consists of the components described below. Note in the description of these components that a basic full-screen system can be implemented using only the protocol, format, and vgroup named PUBLIC. Additional features for a complete interface, such as a help system and printing functions, can be added in a modular fashion after the basic system is working. Source code for all components of NEWSPAPER INDEX is included in the appendices.

FILES:

- NEWSPAPERINDEX is the SPIRES file for the database of newspaper articles. It consists of two subfiles. The subfile NEWSPAPER INDEX is the database of articles. The subfile NEWSPAPER SUBJECTS is an authority file of subject headings that are used to provide enhanced subject access to the article database. A single keyword index includes words from the article headlines and from the subject headings assigned by the indexers.
- NPPROTOS is the SPIRES file containing protocols used by NEWSPAPER INDEX. It consists of one subfile named NEWSPAPER PROTOCOLS. All protocols used by the system are executed from this file. A protocol subfile can be generated automatically by SPIRES through the command PERFORM BUILD PROTOCOLS.
- NPHELP is the SPIRES file containing text of help messages. It consists of the single subfile NEWSPAPER HELP. If a help system is not required, then this file is not needed.

PROTOCOLS:

- PUBLIC is the main protocol controlling the interaction of the user with the full-screen interface. This protocol executes the HELP and PRINT protocols if those functions are included in the system. PUBLIC is executed by a SELECT-COMMAND statement in the SUBFILE section of the NEWSPAPER INDEX file definition. The user sees the full-screen interface from the time NEWSPAPER INDEX is first selected for use.
- HELP is the protocol that controls the online help menus and the display of help messages. It is executed by the PUBLIC protocol when the user presses PF1 while entering keywords on the search screen or while viewing the results of a search. If a help system is not required, then this protocol is not needed.
- PRINT is the protocol that controls printing of a result. It is executed by the PUBLIC protocol when the user presses PF9 while viewing the results of a search. If printing is not required, then this protocol is not needed.
- ERRORS is the protocol that displays custom messages. It is executed by the PUBLIC protocol in response to failure of the FIND command. If custom error messages are not required, then this protocol is not needed.

FORMATS:

- PUBLIC is the format that creates and reads screens to get keywords from the user and to display the results of a search.
- MSG is the format that creates screens that display welcome messages when NEWSPAPER INDEX is selected and an exit message when it is exited. If such messages are not required, then this format is not needed.

- HELP is the format that creates the help menu screens, gets the user's selection, and displays the help information. If a help system is not required, then this format is not needed.
- PRINT is the format that controls the appearance of printouts. If printing is not required, then this format is not needed.

VGROUPS:

- PUBLIC defines variables used by the PUBLIC protocol and format.
- HELP defines variables used only by the help subsystem.
- PRINT defines variables used only by the print subsystem.

Note again that a basic full-screen interface for a SPIRES subfile can be created using only the PUBLIC protocol, format, and vgroup in conjunction with a protocol subfile.

PUBLIC: A DATABASE-INDEPENDENT PROTOCOL

The PUBLIC protocol that controls the full-screen interface for NEWSPAPER INDEX has been written so as to be almost completely independent of any particular SPIRES subfile. As a result, the PUBLIC protocol can be used with any application that meets the following two conditions:

- SPIRES formats for the application must conform to certain naming conventions
- The data to be displayed for each record can be displayed on a single screen.

This section describes the logic followed by each procedure in the PUBLIC protocol that controls the full-screen interface. The reader should refer to the source code in Appendix 1. The description will discuss each procedure in turn as listed in the source code.

MAIN

The MAIN procedure consists of only 35 lines of code. This procedure initializes the session when the subfile is selected, displays a welcome message, sets the PUBLIC format which controls the search and result screens and then creates a blank search screen by executing the procedure SRCH.SCRN. MAIN then enters a loop that executes indefinitely until the user chooses to exit from the application.

Upon entering the loop, the SRCH procedure is executed to display the search screen and to determine which of four actions the user wishes to take. If the user's choice of action is HELP, the contents of the search screen are stored for future use and the HELP protocol is invoked. The parameter S is passed to the HELP protocol to indicate that the user is in search mode. Upon returning from HELP, the search screen is restored and control returns to the beginning of the loop to wait for the user's next action.

If the user's choice of action is FIND, then the procedure SRCH.GETARG is executed to read the keywords from the search screen and to construct a search argument. If no keywords were entered by the user so that the SrchArg variable is null, then the loop is iterated. Otherwise, the procedure SRCH.DOFIND is executed to conduct the search of the subfile. If there is no result, or if the number of records found exceeds the limit in the variable MaxResult, then the loop is reentered and the user is returned to the search screen and presented with an informative message. (Note that the search screen will be restored showing the keywords previously entered so that the user may simply edit the existing keywords rather than reentering the entire search.) If a valid result is found, then the results are sequenced according to the argument in the variable SequenceArg. The procedure RSLT.STOR.KEYS is executed to save the keys of retrieved records in an array, and the procedure RSLT is executed to display the results to the user. When the user finishes reviewing the results and returns from the RSLT procedure, a message calling attention to important function keys is overlaid

on the COMMAND area. The loop is then iterated. This redisplays the search screen containing the keywords that resulted in the successful search, along with the new message about the function keys.

If the user's choice of action is AGAIN, the RSLT procedure is executed to redisplay an existing result.

If the user's choice is EXIT, then the loop variable is set to FALSE, resulting in termination of the loop. The format is set to MSG, the frame GOODBYE is executed to display an exit message, and the system exits from SPIRES.

INITIALIZE SESSION

This procedure is executed once by MAIN when the system is started. First, it sets various SPIRES system parameters. Then it defines four logical "areas" used by SPIRES to buffer data before the data is written to the user's CRT device. (The manual *SPIRES Device Services* explains SPIRES' use of areas to buffer data being transferred between physical devices. Readers unfamiliar with the concept may simply think of an area as a physical location on the terminal screen.) The area COMMAND is 5 lines high by 79 columns wide. It is used to hold messages to the user and to display information such as PF key definitions and screen titles. It is mapped to the CRT screen to begin at line 1, column 1. The area MSG2USER is used to hold messages to the user. It is mapped to the third line of the CRT so that it overlaps the data written to the CRT from the COMMAND area. The area RECORD is 19 lines long and is used to hold the contents of the search screen and the output from the result screen. It is mapped to display on the CRT starting on line 6, just below the area COMMAND. Finally, the area FULLCRT is mapped to fill the entire physical screen for the display of the opening logo.

The procedure then allocates the global vgroup PUBLIC to make available the variables that will be used in the application. Note that the allocate command is specific to the application because the ID of the VGROUP contains the name of a userid, in this case NEWSPAPR. It would be possible to eliminate this dependence on the application if it was convenient to define vgroups on a userid separate from the userid owning the application in question.

Finally, the help subfile is selected and a format is set. This part of the procedure is also database specific because a database specific help file is being used. A generic help subfile could be created and shared by many applications, but it is easier to maintain help records for a particular application by storing them in a dedicated help subfile.

RSLT

This procedure responds to the user's actions while viewing the results of a search. When executed, the existing search screen with the user's keywords is stored. The text of messages and labels appearing in the COMMAND area are assigned to variables, and the COMMAND area is refreshed by execution of the frame CMDOUT in the PUBLIC format. The RSLT.SCRN procedure is then executed to copy the currently selected record in the result to the RECORD area and to update the message describing which record in the result is being viewed. RSLT then enters a loop that executes indefinitely.

Upon entering the loop, the physical terminal screen is refreshed by the command WRITE CRT. This copies the contents of the logical areas to the physical screen and parks the cursor out of the way at the bottom of the

screen. The system then waits for the user to press a function key. The SPIRES function $getarea(crt,fkey) sets the value of the variable CmdNum depending on which function key is pressed by the user.

If the Enter key is pressed (CmdNum = 0), the loop is iterated and no action is taken.

If PF1 is pressed (CmdNum = 201), the existing result screen is stored and the help system is invoked with the parameter D to indicate help is wanted with the display of results. Upon returning from the help system, the result screen is redisplayed and the loop is iterated.

If PF3 is pressed, the RECORD area is blanked and control passes out of the loop. The area CRT is restored so that the search screen and keywords are again available for display, and the procedure returns to the calling procedure MAIN. Note that if the RECORD area is not first blanked, then some data from the search result may show through on to the restored CRT area in locations where the restored area does not contain information.

If PF7 is pressed, the previous record in the sequenced result is displayed. The variable Index, which is the counter for the array holding the keys of the retrieved records, is decreased by 1, and the procedure RSLT.SCRN is executed to display the record. Note that when the record being viewed is the first record in the stack, pressing PF7 causes the system to wrap around to display the last record in the result.

If PF8 is pressed, the next record in the result is displayed by adjusting the value of the variable Index. When the currently viewed record is the last record in the stack, pressing PF8 causes the system to wrap around to display the first record in the stack.

If PF9 is pressed, the protocol PRINT is executed to allow the user to print the current result.

RSLT.SCRN

This procedure creates a message to the user indicating which record in a result is being viewed. It then copies that record to the area RECORD using the frame SRCHRESULT in the format PUBLIC.

RSLT.STOR.KEYS

This procedure fills the array ResultKey with the values of the keys of the records in the stack. Note that counting begins with Index = 1 rather than with the SPIRES default array index of 0. This means the first element of the array is not used. Hence, the array has to be defined as having one element more than the number of records in the largest allowed result.

SRCH

This procedure responds to the user's actions while keywords are being entered on the search screen. It is executed by the procedure MAIN with the sole purpose of setting a value for the variable Action.

When executed, SRCH enters a loop and writes the contents of the area CRT (in this case the search screen data entry form) to the physical terminal screen and locates the cursor at the desired starting position for data entry. The user is then able to enter keywords on the terminal screen before pressing a function key. The SPIRES function $getarea(crt,fkey) sets the value of the variable CmdNum depending on which function key is pressed by the user.

If the Enter key (CmdNum = 0) is pressed, the loop is iterated and no action is taken.

If PF1 is pressed (CmdNum = 201), the variable Action is set to HELP and control returns to MAIN.

If PF2 is pressed, Action is set to FIND, and control returns to MAIN.

If PF3 is pressed, the SRCH.SCRN procedure is executed to create a new blank search screen. This has the effect of erasing any keywords so that a completely new search can be started. A message is created for the user, and control is returned to MAIN.

If PF4 is pressed, the system checks to see if there is an existing result by examining the value of the variable ResultKeyCount, which is the number of keys stored in the array ResultKey. If no result exists for viewing, a message is created for the user and control returns to MAIN. If a result does exist, the variable Action is set to AGAIN. Control returns to MAIN, which redisplays the existing result. Note that the most recently obtained result can be viewed even if a subsequent search fails to obtain a new result.

If PF10 is pressed, Action is set to EXIT, and control is returned to MAIN.

SRCH.DOFIND

This procedure issues the SPIRES command FIND and performs validity tests on the outcome. The procedure first clears any existing SPIRES result so that if the FIND command fails then subsequent tests of the SPIRES variable $result will be valid. (Note that keys of the records found in the most recent successful search are still stored in the ResultKey array. This allows redisplay of the results of that successful search.) It then issues the FIND command using the search argument in the variable SrchArg. If the FIND command fails, then the SPIRES system error code and message numbers are copied to variables and the ERRORS protocol is executed to display an appropriate explanation to the user. The value of the SPIRES system variable $result is then tested, an appropriate message is created, and control is returned to the calling procedure. Note that a search that fails due to invalid syntax will still display the message "No articles found...." The ERRORS protocol is used to display a more informative message. Note also that the text of Msg2User could be made more generic by beginning the phrase with the words "No records...." Alternatively, a variable could be defined to contain an appropriate word to replace the word "articles" so that the protocol would be more independent of the subfile in use.

SRCH.GETARG

This procedure obtains the search argument created from the keywords entered on the search screen. It does this by executing the frame SRCHFORMGET from the format PUBLIC in the area RECORD. SRCHFORMGET reads the area RECORD, which contains the contents of the CRT screen. In the process, it builds the string variable SrchArg by concatenating the keywords entered by the user with the names of appropriate indexes. (SRCHFORMGET is the only frame in the PUBLIC format containing a significant amount of procedural code.) If the user did not enter any keywords then the procedure creates an appropriate message to the user before returning control.

SRCH.SCRN

This procedure sets the value of variables that control the content of messages on the search screen. It then writes the information to the COMMAND area by executing the frame CMDOUT in the format PUBLIC. Finally, it executes the frame SRCHFORMDISP in the PUBLIC format to create the data entry form in the RECORD area.

WELCOME

This procedure controls the welcoming screens by setting the format to MSG and executing frames that display the messages.

FORMATS

The PUBLIC protocol described in the previous section is written in a manner that is almost completely independent of the subfile with which it will be used. Virtually all subfile-specific code for the full-screen interface is in the SPIRES format PUBLIC, which creates the screen displays and transfers data between SPIRES and the physical devices. Fortunately, the format code is almost entirely descriptive and non-procedural. The system designer needs only to describe the location of data elements and associated screen labels. A small amount of procedural code in the SRCHFORMGET frame of the format is used to build the search argument from the keywords entered on the search screen by the user.

A discussion of SPIRES formats sufficient to explain everything involved in the full-screen interface is far beyond the scope of this document. The following conceptual outline focuses on concepts specific to customizing a format to work with the protocol PUBLIC. The discussion should provide the beginner with a focus for approaching the documentation in the manual *SPIRES Formats*

One whose experience with SPIRES formats is limited to using $REPORT to format a screen or printout may naturally think of a format as a means to arrange data on the screen or on a printout in an attractive and convenient way. More importantly, however, SPIRES formats provide a method for moving data among the physical devices of the computer system. A format is used for getting data from a SPIRES subfile and arranging it for presentation on the screen. Formats also are used to read data from the computer screen and to pass data to other SPIRES subsystems.

Most of the code needed to create a functioning search screen, and all of the code necessary to create the result screen, is purely descriptive and non-procedural. Although the FORMATS language is highly formalized, almost all of the code that gets information from the user and displays data to the user can be understood by non-programmers. The small amount of procedural code necessary to construct a search argument from the data entered on the search screen is shown in Appendix 3 in the source code for the frame SRCHFORMGET. Generalizing the code for a search screen more complex than that used for NEWSPAPER INDEX can be done in an obvious, straightforward way.

A skeletal outline for the PUBLIC format to be used with the PUBLIC protocol can be constructed without discussing the details of the data elements in a particular subfile.

First, note that a SPIRES format is divided into frames. Each frame is associated with a particular function or action, such as getting data from a screen or getting data from the subfile and displaying it on the screen. Each frame in a format is further divided into label-groups. Each label-group has a particular function or performs a particular action within its frame. A single label-group may read keywords from particular line on a data entry form, or it may display the value of a single data element at a particular position on the result screen. In each case, the code in a label-group is almost solely descriptive rather than procedural.

Below is the functional skeleton of the SPIRES format PUBLIC as used in NEWSPAPER INDEX. The important feature of this outline is the naming conventions for the frames. These naming conventions are those used by the PUBLIC protocol when it executes or uses frames to create the screen displays. Full details for this format as customized for use with NEWSPAPER INDEX are in Appendix 3. Study of that format will demonstrate to the reader how to write procedural code in each label group of the SRCHFORMGET frame to build a search argument that can be used with the SPIRES command FIND.

```
ID = userid:PUBLIC;
 FILE = userid:SpiresFileName;
 RECORD-NAME = REC01;
 FRAME-ID = CMDOUT;
   COMMENTS = Writes complete command area, top of screen;
   DIRECTION = OUTPUT;
   FRAME-DIM = NumberOfRows,NumberOfColumns;
   USAGE = ALL, NAMED;
   ...label groups...
 FRAME-ID = MSGOUT;
   COMMENTS = Writes new message to the area MSG2USER;
   DIRECTION = OUTPUT;
   FRAME-DIM = NumberOfRows,NumberOfColumns;
   USAGE = ALL, NAMED;
   ...label groups...
 FRAME-ID = SRCHFORMDISP;
   COMMENTS = Writes blank search screen to be filled in by user;
   DIRECTION = OUTPUT;
   FRAME-DIM = NumberOfRows,NumberOfColumns;
   USAGE = ALL, NAMED;
   ...label groups...
 FRAME-ID = SRCHFORMGET;
   COMMENTS = Reads screen to get keywords. Builds search argument;
   DIRECTION = INOUT;
   FRAME-DIM = NumberOfRows,NumberOfColumns;
   USAGE = NAMED;
   ...label groups...
 FRAME-ID = SRCHRESULT;
   COMMENTS = Displays search results;
   DIRECTION = OUTPUT;
   FRAME-DIM = NumberOfRows,NumberOfColumns;
   ...label groups...
 FORMAT-NAME = PUBLIC;
   ALLOCATE = NEWSPAPR:PUBLIC;
   FRAME-NAME = CMDOUT;
     FRAME-TYPE = XEQ;
   FRAME-NAME = MSGOUT;
     FRAME-TYPE = XEQ;
   FRAME-NAME = SRCHFORMGET;
     FRAME-TYPE = XEQ;
   FRAME-NAME = SRCHFORMDISP;
     FRAME-TYPE = XEQ;
   FRAME-NAME = SRCHRESULT;
     FRAME-TYPE = DATA;
```

This skeletal outline can be used with any SPIRES subfile to create a full-screen interface. Generically chosen names for the format, frames, and global variable group are independent of the SPIRES subfile. Only the elements of each label group

are dependent on the subfile; for they must refer to the names of data elements in the subfile.

The format is named PUBLIC. The frame CMDOUT describes the locations and characteristics of text to be written to the command area of the screen. In NEWSPAPER INDEX the command area constitutes the top five lines of the screen. It is used to display a title for the screen, instructions and messages to the user, a list of function keys, and a divider between these prompts and the portion of the screen used for data entry and display.

The frame MSGOUT is used to display a message to the user. In NEWSPAPER INDEX, the message is written to the third line of the screen, overlapping the area COMMAND. The frame SRCHFORMDISP creates the full-screen form on which the user enters keywords for a search. The frame SRCHFORMGET reads the screen to get the keywords, and it builds the search argument passed by the PUBLIC protocol to the SPIRES command FIND. The frame SRCHRESULT displays the result of the search.

ADAPTING THIS SYSTEM

The appendices to this paper contain complete source code for the NEWSPAPER INDEX application as it was working in the Virginia Tech Libraries on 9/24/92 using SPIRES version 89.03 under IBM's CMS operating system. A copy of the code may be obtained from the author by sending a request to KRIZ@VTVM1.BITNET or KRIZ@VTVM1.CC.VT.EDU.

A basic, functioning full-screen interface can be devised for any SPIRES subfile using the PUBLIC protocol and PUBLIC vgroup almost without modification in conjunction with a subfile-specific PUBLIC format which follows the required naming conventions. Note that the file definition for the protocol subfile is not included because such a subfile can be generated by a SPIRES system utility invoked by the PERFORM BUILD PROTOCOLS command.

The steps below may be followed in applying the PUBLIC protocol to an existing SPIRES subfile. Code for the PF1 key to invoke the help system, the PF9 key to invoke the print protocol, and the code for error testing should be commented out of PUBLIC protocol until after the search and display systems are functioning.

1. Design a search screen for entering keywords corresponding to the indexes in the subfile.
2. Design a result screen to display records retrieved in a search.
3. Add the vgroup PUBLIC as shown in Appendix 2 to the VGROUPS subfile and compile.
4. Write a format named PUBLIC to create the screens for the subfile. Follow the required naming convention for frames in the format so they can be used and executed by the PUBLIC protocol. Add the format to the FORMATS subfile and compile.
5. Create a protocol subfile using the PERFORM BUILD PROTOCOLS command.
6. Add the protocol PUBLIC as shown in Appendix 1 to the protocol subfile.
7. Select the subfile with the data and use the SET XEQ command to activate the protocol subfile.
8. Issue the command ..PUBLIC.
9. Customize and add other modules, including HELP, PRINT, and ERRORS.

APPENDIX 1. PUBLIC PROTOCOL

The workings of this protocol are fully described on page 15. In reading the code, note that there are almost no references that are specific to the NEWSPAPER INDEX application. The exceptions are the lines in the INITIALIZE.SESSION procedure that allocate the vgroup NEWSPAPR:PUBLIC and select the subfile NEWSPAPER HELP. All procedures, variables, and format frames are generically named. However, the formats for NEWSPAPER INDEX must be written to conform to the naming conventions used in this protocol. The shading indicates a single line of code that has wrapped to a second physical line on the page because it does not fit between the page margins. Such lines should be entered as single lines in a program.

```
* PUBLIC (Add 08/27/91, Upd 09/24/92 at 15:31 by NEWSPAPR)
!set noecho
- ************************************************************************
- SEE DOCUMENTATION AND USAGE NOTES AT END OF FILE
-
- IMPORTANT:  Check commenting of ++INITIALIZE and attn='-' before
-            installing for public use
- ************************************************************************
- ========================= MAIN PROGRAM BEGINS ==========================
++MAIN
xeq proc INITIALIZE.SESSION          ; Initialize SPIRES and variables
xeq proc WELCOME                     ; Display welcome messages
set format PUBLIC                    ; Controls search and display
xeq proc SRCH.SCRN                   ; Create the search screen

WHILE #SeeMore

   xeq proc SRCH                     ; Get search argument and/or action

   if #Action = 'HELP' then beginblock          ; Get help for searching
                       store area CRT
                       ..HELP S
                       restore area CRT
                       iterate
                       endblock
   if #Action = 'FIND' then beginblock
                       xeq proc SRCH.GETARG     ; Get search argument,
                                                ; set msg if null
                       if #SrchArg = '' then iterate
                       xeq proc SRCH.DOFIND     ; Search, test result,
                                                ; set msg if out of range
                       if $result = 0 or > #MaxResult then iterate
                       /sequence #SequenceArg   ; Sort records
                       let SrchArgOK = #SrchArg
                       xeq proc RSLT.STOR.KEYS  ; Put keys of result in array
                       xeq proc RSLT            ; View result

                       let Msg2User = 'MODIFY SEARCH:  Edit keywords, press PF2   --   NEW
SEARCH:  Press PF3'
                       in MSG2USER, xeq frame MSGOUT
                       iterate
                       endblock
    if #Action = 'AGAIN' then beginblock        ; View existing result
                         xeq proc RSLT
                         iterate
                         endblock
    if #Action = 'EXIT' then let SeeMore = $FALSE
                        then iterate

ENDWHILE
```

```
set format MSG
in RECORD, xeq frame GOODBYE

EXIT QUIET                                          ; leave SPIRES

- ========================= MAIN PROGRAM ENDS ==========================

- ========================= PROCEDURES BEGIN ===========================
-----------
++INITIALIZE.SESSION

set messages 0   ; No SPIRES messages will be sent to terminal
set nowrite      ; Turn off automatic writing when an area is full
set noread       ; Turn off automatic reading when an area is full
set nostop       ; Command failure won't stop execution.
                 ; Check for failure with traps in protocol as necessary

define area COMMAND (5,79) on CRT(1,1) bgprotect    ; prompts, etc
define area MSG2USER (1,79) on CRT(3,1) bgprotect   ; message updates
define area RECORD (19,79) on CRT(6,1) bgprotect    ; search form & result
define area FULLCRT (24,79) on CRT(1,1) bgprotect   ; for greetings

allocate NEWSPAPR:PUBLIC                    ; Global VGROUP

through HELPPATH select NEWSPAPER HELP      ; Select the help subfile
through HELPPATH set format HELP            ; Set the help format

RETURN

-----------
++RSLT
- Display search results

store area CRT              ; Save so can redisplay keywords for
                            ; subsequent modification

let ScrnTitle =  #RsltScrnTitle
let Instruction = 'FIND: ' #SrchArgOK
let FnKeys = 'PF1=Help  PF3=Quit viewing  PF7=Previous article  PF8=Next article
PF9=Print'
in COMMAND, xeq frame CMDOUT

xeq proc RSLT.SCRN                          ; Display first record

WHILE #Forever
   write CRT, read cursor(24,1)   attn='-'   ; Write results to CRT, park cursor out of
way
   let CmdNum = $getarea(crt,fkey)          ; Get function key
   if #CmdNum =   0 then iterate            ; <Enter> no action
   if #CmdNum = 201 then beginblock         ; <PF1> Help for display of records
                    store area CRT
                    ..HELP D
                    restore area CRT
                    iterate
                    endblock
   if #CmdNum = 203 then blank RECORD       ; Quit displaying results
                    then leave
   if #CmdNum = 207 then beginblock         ; Display previous record
                    if #Index > 1 then let Index = #Index -1
                    else let Index = #ResultKeyCount
                    xeq proc RSLT.SCRN      ; Update msg & record
                    iterate
                    endblock
   if #CmdNum = 208 then beginblock         ; Display next record
                    if #Index < #ResultKeyCount then let Index = #Index + 1
                    else let Index = 1
                    xeq proc RSLT.SCRN
                    iterate
                    endblock
   if #CmdNum = 209 then beginblock         ; Print the result
                    ..PRINT
```

```
                    iterate
                    endblock

ENDWHILE

restore area CRT          ; Restore search screen with keywords that
                          ; generated result, but with new Msg2User

RETURN

----------------
++RSLT.SCRN
- Refresh message and record areas during display of result

if #ResultKeyCount = 1: let Msg2User = 'Record ' #Index ' of ' #ResultKeyCount '  --Press
PF3 to quit'
else if #Index < #ResultKeyCount: let Msg2User = 'Record ' #Index ' of ' #ResultKeyCount
     else let Msg2User = 'LAST RECORD:  Record ' #Index ' of ' #ResultKeyCount '  --Press
PF3 to quit'
      else eval $getarea(crt,bell)

in MSG2USER, xeq frame MSGOUT
/in RECORD, using SRCHRESULT, display #ResultKey::Index

RETURN

-----------
++RSLT.STOR.KEYS
- Fill array with keys to search result starting at Index = 1

let ResultKeyCount = $stack
for stack
let Index = 0
WHILE Index < $stack
   let Index = #Index + 1
   in null show key next    end = 'leave'
   let ResultKey::Index = $key
ENDWHILE
let Index = 1

RETURN

-----------
++SRCH
-  Display search screen and return search argument and/or action

WHILE #Forever
  /write CRT, read cursor(#CrsrSrchRow,#CrsrSrchCol)    attn='-'
   let CmdNum = $getarea(crt,fkey)                ; Get function key
   if #CmdNum =   0 then iterate                  ; <Enter> No action
   if #CmdNum = 201 then let Action = 'HELP'      ; <PF1>
                    then leave
   if #CmdNum = 202 then let Action = 'FIND'
                    then leave
   if #CmdNum = 203 then let Msg2User = 'Words erased. PF4 shows an existing result,  PF10
exits system.'
                    then xeq proc SRCH.SCRN       ; Rebuild search scrn
                    then iterate
   if #CmdNum = 204 then beginblock
                    if #ResultKeyCount = 0 then beginblock
                                              let Msg2User = 'No result to display'
                                              eval $getarea(crt,bell)
                                              in MSG2USER, xeq fram MSGOUT
                                              iterate
                                              endblock
                    else beginblock
                         let Action = 'AGAIN'
                         leave
                         endblock
                    endblock
   if #CmdNum = 210 then let Action = 'EXIT'
                    then leave
```

```
ENDWHILE

RETURN

-----------
++SRCH.DOFIND
- Find and test search result
 clear result                ; So if command fails, $result will be null
                             ; and not value from previous result
/find #SrchArg

 if $NO then beginblock              ; Test for failed FIND command
        let ErrMsgNum = $msgnum      ; Store the SPIRES message numbers
        let ErrSNum = $snum
        let ErrENum = $enum
        ..ERRORS                     ; Display an error message
        endblock

 if $result = 0 then let Msg2User = 'No articles found for ' #SrchArg
 if $result > #MaxResult then let Msg2User = $result ' articles retrieved. Please narrow
your search.'
 if $result = 0 or $result > #MaxResult then eval $getarea(crt,bell)
                                 then in MSG2USER, xeq frame MSGOUT

RETURN

-----------
++SRCH.GETARG
- Get the search argument

in RECORD, xeq frame SRCHFORMGET        ; Get search argument from screen

if #SrchArg = '' then beginblock        ; Check for null SrchArg
                 let Msg2User = 'You must enter a keyword or date'
                 eval $getarea(crt,bell)
                 in MSG2USER, xeq frame MSGOUT
                 endblock
RETURN

-----------
++SRCH.SCRN
- Reset the screen display for searching

/let ScrnTitle = '#SrchScrnTitle'           ; Assign variables
 let Instruction = 'Enter keywords, limit dates, then press PF2'
 let FnKeys = 'PF1=Help  PF2=Do search  PF3=Erase words  PF4=Show again          PF10=Exit'
 in COMMAND, xeq frame CMDOUT               ; Create command area
 in RECORD, xeq frame SRCHFORMDISP          ; Create blank search form

RETURN

-----------
++WELCOME

set format MSG                      ; MSG contains text screens
in FULLCRT, xeq frame LOGO          ; Display large title screen

ASK AREA FULLCRT 24,1 PROMPT = 'Press <ENTER> to continue' attn='-'    ; Wait for <ENTER>

in FULLCRT, xeq frame GREETING      ; Say hello, wait for <ENTER> key

RETURN
- ========== PROCEDURES END ==========

- ========== DOCUMENTATION AND USAGE NOTES BEGIN ==========
- PUBLIC protocol controls the full screen application of the SPIRES
-        database NEWSPAPER INDEX.
-        PA1 (esc Z) aborts to CP. CLEAR (ATTN) key has no effect
-
- Written by: H. M. Kriz, University Libraries, 231-7052, KRIZ@VTVM1
- Called by:  SELECT-COMMAND in public access subfile NEWSPAPER INDEX
- Calls:
```

```
-    SUBFILES:  NEWSPAPER HELP - Text for help screens
-               NEWSPAPER PROTOCOLS - Set xeq by SELECT-COMMAND in NEWSPAPER INDEX
-    FORMATS for subfile NEWSPAPER INDEX
-        MSG     - Displays text screens on entry & exit
-        PUBLIC - Controls search, display
-    PROTOCOLS executed by PUBLIC protocol from NEWSPAPER PROTOCOLS subfile
-        ERRORS - Displays custom error messages
-        HELP    - Controls menus and text display for NEWSPAPER HELP
-        PRINT   - Controls printing of a search result
-    VGROUPS allocated by PUBLIC protocol
-        PUBLIC - All variables used in PUBLIC protocol and PUBLIC format
-
- Variables used in PUBLIC protocol compiled in vgroup NEWSPAPR:PUBLIC
-    Action          - instruction returned by function key menu
-    CmdNum          - value returned for function key by $getarea(crt,fkey)
-    CrsrSrchRow     - starting row of cursor on search screen
-    CrsrSrchCol     - starting column of cursor on search screen
-    ErrENum         - set to value of $enum following error
-    ErrMsgNum       - set to value of $msgnum following error
-    ErrSNum         - set to value of $snum following error
-    FnKeys          - text listing PFkey actions
-    Forever         - loop variable
-    Index           - index counter for ResultKey array
-    Instruction     - text telling user what action to take on a screen
-    MaxResult       - maximum number of records allowed in a result
-    Msg2User        - message displayed in command area
-    ResultKey       - array of keys for the search result
-    ResultKeyCount  - number of record keys in ResultKey array
-    RsltScrnTitle   - text of title for result screen
-    ScrnTitle       - screen title displayed in COMMAND area
-    SeeMore         - loop variable
-    SequenceArg     - argument for sequence command
-    SrchArg         - argument for FIND command
-    SrchArgOK       - search argument which last produced a result
-    SrchScrnTitle   - text of title for search screen

- -----------
-  HISTORY:   Adapted from MEDIA system, 8/27/91 by H. Kriz
-  Modified:  9/21-24/92 by H. Kriz to use generic names and variables
-
- ================ DOCUMENTATION AND USAGE NOTES END ================
```

APPENDIX 2. PUBLIC VGROUP

```
VGROUP = NEWSPAPR:PUBLIC;
COMMENTS = ================================================================;
COMMENTS = For use with the PUBLIC format and PUBLIC protocol for      -;
COMMENTS = NEWSPAPER INDEX                                             -;
COMMENTS = ================================================================;
AUTHOR = H. M. Kriz, University Libraries, 231-7052;
MODDATE = THUR. SEPT. 24, 1992;
DEFDATE = THUR. OCT. 24, 1991;
VARIABLE = Action;
  LENGTH = 5;
  TYPE = STRING;
  COMMENTS = Instruction returned by function key menu;
VARIABLE = CmdNum;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Value returned for function key by $getarea(crt,fkey);
VARIABLE = CrsrSrchCol;
  LENGTH = 1;
  TYPE = INT;
  COMMENTS = Starting column for the cursor on the search screen;
  VALUE = 5;
VARIABLE = CrsrSrchRow;
  LENGTH = 1;
  TYPE = INT;
  COMMENTS = Starting row for the cursor on the search screen;
  VALUE = 7;
VARIABLE = DateArg;
  LENGTH = 32;
  TYPE = STRING;
  COMMENTS = Date portion of search argument;
VARIABLE = ErrENum;
  TYPE = INT;
  COMMENTS = Set to value of $enum when error occurs;
VARIABLE = ErrMsgNum;
  TYPE = INT;
  COMMENTS = Set to value of $msgnum when error occurs;
VARIABLE = ErrSNum;
  TYPE = INT;
  COMMENTS = Set to value of $snum when error occurs;
VARIABLE = FnKeys;
  LENGTH = 79;
  TYPE = STRING;
  COMMENTS = Function key list in command area;
VARIABLE = Forever;
  TYPE = FLAG;
  COMMENTS = Loop variable;
  VALUE = $TRUE;
VARIABLE = Index;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Index variable for ResultKey array;
VARIABLE = Instruction;
  LENGTH = 79;
  TYPE = STRING;
  COMMENTS = Text in instruction line of command area;
VARIABLE = MaxResult;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Maximum size of search result. Value is 1 less than array size;
  VALUE = 250;
VARIABLE = Msg2User;
  LENGTH = 79;
  TYPE = STRING;
  COMMENTS = Message sent to user;
VARIABLE = ResultKey;
  OCCURS = 251;
```

```
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Array to hold keys of result stack. Occurs MaxResult + 1;
  INDEXED-BY = Index;
VARIABLE = ResultKeyCount;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Number of record keys in ResultKey array;
VARIABLE = RsltScrnTitle;
  LENGTH = 20;
  TYPE = STRING;
  COMMENTS = Title appearing on result screen;
  VALUE = 'ARTICLES';
VARIABLE = ScrnTitle;
  LENGTH = 20;
  TYPE = STRING;
  COMMENTS = Text for screen title in command area;
VARIABLE = SeeMore;
  TYPE = FLAG;
  COMMENTS = Loop variable;
  VALUE = $TRUE;
VARIABLE = SequenceArg;
  LENGTH = 20;
  TYPE = STRING;
  COMMENTS = Argument for sequence command;
  VALUE = 'date(d) title';
VARIABLE = SrchArg;
  LENGTH = 400;
  TYPE = STRING;
  COMMENTS = Search argument built from reading screen;
VARIABLE = SrchArgOK;
  LENGTH = 400;
  TYPE = STRING;
  COMMENTS = Search argument which last produced a result;
VARIABLE = SrchScrnTitle;
  LENGTH = 20;
  TYPE = STRING;
  COMMENTS = Title appearing on search screen;
  VALUE = 'NEWSPAPER SEARCH';
```

APPENDIX 3. PUBLIC FORMAT

```
ID = NEWSPAPR:PUBLIC;
COMMENTS = ================================================================;
COMMENTS = PUBLIC format for fullscreen interface for NEWSPAPER INDEX -;
COMMENTS = Modified 9/21/92 by H. Kriz                                -;
COMMENTS = Names of FORMAT, VGROUP and variables changed to use more  -;
COMMENTS = generic form                                               -;
COMMENTS = ================================================================;
AUTHOR = H. M. Kriz, University Libraries,231-7052;
DEFDATE = WED. SEPT. 4, 1991;
MODDATE = MON. SEPT. 21, 1992;
MODTIME = 11:10:20;
FILE = NEWSPAPR:NEWSPAPERINDEX;
RECORD-NAME = REC01;
FRAME-ID = CMDOUT;
  COMMENTS = Writes complete command area, top of screen;
  DIRECTION = OUTPUT;
  FRAME-DIM = 5,79;
  USAGE = ALL, NAMED;
  LABEL = SCRNTITLE;
    VALUE = #ScrnTitle;
    LENGTH = 20;
    START = 1,30;
    UPROC = set adjust center;
    DISPLAY = BRIGHT;
    PUTDATA;
  LABEL = INSTRUCTION;
    VALUE = #Instruction;
    LENGTH = 79;
    START = 2,1;
    PUTDATA;
  LABEL = MSG2USER;
    VALUE = #Msg2User;
    LENGTH = 79;
    START = 3,1;
    DISPLAY = BRIGHT;
    PUTDATA;
  LABEL = FNKEYS;
    VALUE = #FnKeys;
    START = 4,1;
    PUTDATA;
  LABEL = DIVIDER;
    VALUE = '=';
    UPROC = set repeat;
    PUTDATA;
FRAME-ID = MSGOUT;
  COMMENTS = Writes new message to the area MSG2USER. Overlaps COMMAND area;
  DIRECTION = OUTPUT;
  FRAME-DIM = 1,79;
  USAGE = ALL, NAMED;
  LABEL = MSG2USER;
    VALUE = #Msg2User;
    LENGTH = 79;
    START = 1,1;
    DISPLAY = BRIGHT;
    PUTDATA;
FRAME-ID = SRCHFORMDISP;
  COMMENTS = Writes blank search screen to be filled in by user;
  DIRECTION = OUTPUT;
  FRAME-DIM = 19,79;
  USAGE = ALL, NAMED;
  LABEL = PROMPT;
    VALUE = 'Enter subject and title keywords on the lines below:';
    START = 1,1;
    DISPLAY = BRIGHT;
    PUTDATA;
  LABEL;
```

```
    VALUE = '==>';
    START = 2,1;
    DISPLAY = BRIGHT;
    PUTDATA;
    LOOP = 2;
    XSTART = *+1,1;
  LABEL;
    MARGINS = 5,78;
    LENGTH = 219;
    START = 2,5;
    DISPLAY = UNPROTECT;
    PUTDATA;
  LABEL = DATE.BEGIN;
    TSTART = 6,1;
    TITLE = 'EXCLUDE articles published before (mm/dd/yy): ';
    LENGTH = 8;
    START = *,47;
    DISPLAY = UNPROTECT;
    PUTDATA;
  LABEL = DATE.END;
    TSTART = 8,1;
    TITLE = ' EXCLUDE articles published after (mm/dd/yy): ';
    LENGTH = 8;
    START = *,47;
    DISPLAY = UNPROTECT;
    PUTDATA;
FRAME-ID = SRCHFORMGET;
  COMMENTS = Reads screen to get keywords. Builds search argument;
  DIRECTION = INOUT;
  FRAME-DIM = 19,79;
  USAGE = NAMED;
  LABEL;
    UPROC = let SrchArg = '';
    UPROC = let DateArg = '';
  LABEL = KEYWORDS;
    MARGINS = 5,78;
    LENGTH = 219;
    START = 2,5;
    GETDATA;
    INPROC = $squ;
    UPROC = if $cval = '' then jump DATE.BEGIN;
    UPROC = let SrchArg = 'KW ' $cval;
  LABEL = DATE.BEGIN;
    LENGTH = 8;
    START = 6,47;
    GETDATA;
    INPROC = $squ;
    UPROC = if $cval = '' then jump DATE.END;
    UPROC = else let DateArg = 'DATE >= ' $cval;
  LABEL = DATE.END;
    LENGTH = 8;
    START = 8,47;
    GETDATA;
    INPROC = $squ;
    UPROC = if $cval = '' then jump FINISH.ARG;
    UPROC = if #DateArg then BEGINBLOCK;
    UPROC = let DateArg = #DateArg ' and <= ' $cval;
    UPROC = jump FINISH.ARG;
    UPROC = ENDBLOCK;
    UPROC = let DateArg = 'DATE <= ' $cval;
  LABEL = FINISH.ARG;
    UPROC = if #DateArg = '' then RETURN;
    UPROC = if #SrchArg: let SrchArg = #SrchArg ' and ' #DateArg;
    UPROC = else let SrchArg = #DateArg;
FRAME-ID = SRCHRESULT;
  COMMENTS = Displays search results;
  DIRECTION = OUTPUT;
  FRAME-DIM = 19,79;
  LABEL = SOURCE;
    TSTART = 2,1;
    TITLE = 'Source: ';
    GETELEM;
```

```
    START = 2,*+1;
    PUTDATA;
  LABEL = DATE;
    TSTART = *,27;
    TITLE = 'Date: ';
    GETELEM;
    START = *,33;
    PUTDATA;
  LABEL = PAGE;
    TSTART = *,44;
    TITLE = 'Page: ';
    GETELEM;
    START = *,*+1;
    PUTDATA;
  LABEL = TITLE;
    TSTART = 3,2;
    TITLE = 'Title: ';
    GETELEM;
    MARGINS = 9,79;
    MAXROWS = 2;
    START = *,9;
    PUTDATA;
  LABEL = SUBJECT;
    ENTRY-UPROC = set buildsep '.  ';
    ENTRY-UPROC = set buildend .;
    TSTART = *+1,2;
    TITLE = 'Subjects: ';
    GETELEM;
    DEFAULT;
    MARGINS = 12,79;
    MAXROWS = 12;
    START = *,12;
    PUTDATA;
    LOOP;
FORMAT-NAME = PUBLIC;
  ALLOCATE = NEWSPAPR:PUBLIC;
  FRAME-NAME = CMDOUT;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
  FRAME-NAME = MSGOUT;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
  FRAME-NAME = SRCHFORMGET;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
  FRAME-NAME = SRCHFORMDISP;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
    UPROC = set autotab;
    UPROC = set tdisplay = bright;
    UPROC = set display = unprotect;
  FRAME-NAME = SRCHRESULT;
    FRAME-TYPE = DATA;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
    UPROC = set autotab;
    UPROC = set tdisplay = bright;
```

APPENDIX 4. HELP PROTOCOL

```
* HELP (Add 09/03/91, Upd 09/24/92 at 15:55 by NEWSPAPR)
!set noecho
- ------------------------------------------------------------------------
-  HELP protocol for NEWSPAPER INDEX.  Executed by PUBLIC protocol
-  Executes frames in PUBLIC format for COMMAND area
-  in addition to frames in format HELP
-  Written:  9/3/91 by H. M. Kriz, University Libraries
-  Modified: 9/21-24/92 by H. Kriz
-  Changed names and variables to more more generic form
- ------------------------------------------------------------------------

let WhichHelp = $ask            ; S for searching, D for displaying result
                                ; WhichHelp passed as parameter for
                                ; ..HELP issued by PUBLIC protocol

-----------
++HELP

let Msg2User = ''                 ; Clear any previous setting

xeq proc HELP.SCRN                ; Create help menu

WHILE #Forever
   write CRT, read cursor(1,29)       attn='-'
   let cmdnum = $getarea(crt,fkey)
   if #cmdnum =   0 then iterate
   if #cmdnum = 202 then beginblock                    ; Get user's choice
                    in RECORD, through HELPPATH xeq frame HELP.GET

                   /if #WhichHelp = 'S' then let HelpNumS = #HelpNum
                   /if #WhichHelp = 'D' then let HelpNumD = #HelpNum
                   /xeq proc HELP.#WhichHelp.DO        ; Display requested
                                                       ; help or get msg
                    in MSG2USER, xeq frame MSGOUT      ; Update msg line
                                                       ; for invalid choice
                    iterate
                    endblock
   if #cmdnum = 203 then leave                         ; Quit help
ENDWHILE

RETURN

-----------
++HELP.SCRN
- Set up help menu

if #WhichHelp = 'S' then beginblock
                    let ScrnTitle = 'HELP FOR SEARCHING'
                   /let HelpNum = #HelpNumS
                    endblock
if #WhichHelp = 'D' then beginblock
                    let ScrnTitle = 'HELP FOR DISPLAY'
                   /let HelpNum = #HelpNumD
                    endblock
let Instruction = 'Enter the number of the desired topic, press PF2'
let FnKeys = '          PF2=View selection       PF3=Quit HELP'

in COMMAND, xeq frame CMDOUT
/in RECORD, through HELPPATH xeq frame HELP.#WhichHelp.MENU

RETURN

-----------
++HELP.D.DO
- Get help while displaying a search result

- Determine the help topic from the number entered by user
```

```
if #HelpNumD < 1 or #HelpNumD > 4  : beginblock          ; check valid number
                                   let Msg2User = 'Choose a number from 1 to 4'
                                    eval $getarea(crt,bell)
                                    return
                                    endblock
if #HelpNumD = 1 then let HelpTopic = 'GENERAL.D'
if #HelpNumD = 2 then let HelpTopic = 'SOURCE.D'
if #HelpNumD = 3 then let HelpTopic = 'SUBJECT.D'
if #HelpNumD = 4 then let HelpTopic = 'FNKEYS.D'

/through HELPPATH using SHOWHELP, dis #HelpTopic  ; display the help

RETURN

-----------
++HELP.S.DO
- Get help while entering keywords on the search screen

- Determine the help topic from the number entered by user

if #HelpNum = '90' : beginblock                    ; Keyword browsing
                  /browse kw #brkw
                   return
                   endblock

if #HelpNum < 1 or #HelpNumS > 13 : beginblock       ; check valid number
                                  let Msg2User = 'Choose a number from 1 to 13'
                                  eval $getarea(crt,bell)
                                  return
                                  endblock
if #HelpNum = 1 then let HelpTopic = 'GENERAL.S'
if #HelpNum = 2 then let HelpTopic = 'FNKEYS.S'
if #HelpNum = 3 then let HelpTopic = 'KEYWORD.S'
if #HelpNum = 4 then let HelpTopic = 'DATE.S'
if #HelpNum = 5 then let HelpTopic = 'NAMES.S'
if #HelpNum = 6 then let HelpTopic = 'NEWSPAPER.S'
if #HelpNum = 7 then let HelpTopic = 'BOOLEAN.S'
if #HelpNum = 8 then let HelpTopic = 'COMPOUND.S'
if #HelpNum = 9 then let HelpTopic = 'NATURAL.S'
if #HelpNum = 10 then let HelpTopic = 'WILDCARD.S'
if #HelpNum = 11 then let HelpTopic = 'COST.S'
if #HelpNum = 12 then let HelpTopic = 'TECHNICAL.S'
if #HelpNum = 13 then let HelpTopic = 'SPECIAL.S'

/through HELPPATH using SHOWHELP, dis #HelpTopic  ; display the help

RETURN
```

APPENDIX 5. HELP VGROUP

```
VGROUP = NEWSPAPR:HELP;
COMMENTS = For use with NEWSPAPER HELP and the HELP protocol;
COMMENTS = in the NEWSPAPER INDEX application;
AUTHOR = H. M. Kriz, University Libraries,   231-7052;
MODDATE = MON. SEPT. 21, 1992;
DEFDATE = WED. SEPT. 4, 1991;
VARIABLE = BrKW;
  LENGTH = 10;
  TYPE = STRING;
  COMMENTS = String where keyword index browsing begins;
  VALUE = 'A';
VARIABLE = HelpNum;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Number selected from the help menu;
  VALUE = 1;
VARIABLE = HelpNumD;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Saved value of HelpNum from DISPLAY;
  VALUE = 1;
VARIABLE = HelpNumS;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Saved value of HelpNum from SEARCH;
  VALUE = 1;
VARIABLE = HelpTopic;
  LENGTH = 20;
  TYPE = STRING;
  COMMENTS = Key to selected record in NEWSPAPER HELP subfile;
VARIABLE = StartCol1;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = First starting column for output;
VARIABLE = StartCol2;
  LENGTH = 4;
  TYPE = INT;
  COMMENTS = Second starting column for output;
VARIABLE = WhichHelp;
  LENGTH = 1;
  TYPE = STRING;
  COMMENTS = Indicator passed to HELP by PUBLIC;
```

APPENDIX 6. HELP FORMAT

```
ID = NEWSPAPR:HELP;
COMMENTS = ================================================================;
COMMENTS = Formats help screens for NEWSPAPER INDEX                        -;
COMMENTS = Modified 9/21/92 by H. Kriz                                     -;
COMMENTS = Adopted generic FORMAT and VGROUP names                         -;
COMMENTS = ================================================================-;
AUTHOR = H. M. Kriz, University Libraries, 703-231-7052;
DEFDATE = THUR. OCT. 24, 1991;
MODDATE = MON. SEPT. 21, 1992;
MODTIME = 10:33:56;
FILE = NEWSPAPR:NPHELP;
RECORD-NAME = REC01;
FRAME-ID = HELP.D.MENU;
  COMMENTS = Display menu of help topics while displaying result;
  DIRECTION = OUTPUT;
  FRAME-DIM = 19,79;
  USAGE = NAMED;
  LABEL;
    UPROC = let StartCol1 = 6;
    UPROC = let StartCol2 = 36;
    UPROC = if #HelpNum = 0 then let HelpNum = 1;
  LABEL;
    TSTART = 3,#StartCol1;
    TITLE = '1) ';
    VALUE = 'General information';
    START = *,*+1;
    PUTDATA;
  LABEL;
    TSTART = *+2,#StartCol1;
    TITLE = '2) ';
    VALUE = 'Source';
    START = *,*+1;
    PUTDATA;
  LABEL;
    TSTART = *+2,#StartCol1;
    TITLE = '3) ';
    VALUE = 'Subjects';
    START = *,*+1;
    PUTDATA;
  LABEL;
    TSTART = *+2,#StartCol1;
    TITLE = '4) ';
    VALUE = 'Function keys';
    START = *,*+1;
    PUTDATA;
  LABEL = INSTRUCTION;
    TSTART = 1,3;
    TITLE = 'Enter number of topic ==> ';
    VALUE = $STRING(#HelpNum);
    LENGTH = 2;
    START = *,29;
    DISPLAY = UNPROTECT;
    PUTDATA;
FRAME-ID = HELP.S.MENU;
  COMMENTS = Display menu of help topics while in search mode;
  DIRECTION = OUTPUT;
  FRAME-DIM = 19,79;
  USAGE = NAMED;
  LABEL;
    UPROC = let StartCol1 = 6;
    UPROC = let StartCol2 = 36;
    UPROC = if #HelpNum = 0 then let HelpNum = 1;
  LABEL;
    TSTART = 3,#StartCol1;
    TITLE = '1) ';
    VALUE = 'General information';
```

```
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol1;
  TITLE = '2) ';
  VALUE = 'Function keys';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol1;
  TITLE = '3) ';
  VALUE = 'Keywords';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol1;
  TITLE = '4) ';
  VALUE = 'Dates';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol1;
  TITLE = '5) ';
  VALUE = 'Personal names';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol1;
  TITLE = '6) ';
  VALUE = 'Newspapers';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = 3,#StartCol2;
  TITLE = '7) ';
  VALUE = 'Boolean operators';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '8) ';
  VALUE = 'Compound search expression';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '9) ';
  VALUE = 'Natural language searching';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '10) ';
  VALUE = 'Wild card searching';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '11) ';
  VALUE = 'Costs';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '12) ';
  VALUE = 'Technical details';
  START = *,*+1;
  PUTDATA;
LABEL;
  TSTART = *+2,#StartCol2;
  TITLE = '13) ';
  VALUE = 'Special services';
```

```
    START = *,*+1;
    PUTDATA;
  LABEL = BROWSE.KW;
    TSTART = 18,5;
    TITLE = '90) List keywords beginning with: ';
    VALUE = #BrKW;
    LENGTH = 10;
    START = *,39;
    DISPLAY = UNPROTECT;
    PUTDATA;
  LABEL = INSTRUCTION;
    TSTART = 1,3;
    TITLE = 'Enter number of topic ==> ';
    VALUE = $STRING(#HelpNum);
    LENGTH = 2;
    START = *,29;
    DISPLAY = UNPROTECT;
    PUTDATA;
FRAME-ID = HELP.GET;
  COMMENTS = Read choices entered by user on either HELP menu;
  DIRECTION = INPUT;
  FRAME-DIM = 19,79;
  USAGE = MAINT;
  LABEL;
    LENGTH
    START
    GETDATA;
    INPROC = $squ;
    INPROC = if ~ $TYPETEST($cval,INT) then set cval = '0';
    UPROC = let HelpNum = $cval;
    UPROC = if #WhichHelp = 'D' then jump ALL.DONE;
  LABEL = BROWSE.KW;
    LENGTH = 10;
    START = 18,39;
    GETDATA;
    INPROC = $squ;
    UPROC = let brkw = $cval;
  LABEL = ALL.DONE;
FRAME-ID = SHOWHELP;
  COMMENTS = Display text of the help message;
  DIRECTION = OUTPUT;
  FRAME-DIM = 0,79;
  LABEL = HELPTITLE;
    GETELEM;
    LENGTH = 79;
    START = 1,1;
    UPROC = set adjust center;
    DISPLAY = BRIGHT;
    PUTDATA;
  LABEL = HELPTEXT;
    GETELEM;
    MARGINS = 5,75;
    START = *+2,5;
    PUTDATA = 2;
    LOOP;
    XSTART = *+1,5;
FORMAT-NAME = HELP;
  ALLOCATE = NEWSPAPR:PUBLIC;
  ALLOCATE = NEWSPAPR:HELP;
  FRAME-NAME = HELP.D.MENU;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
    UPROC = set tdisplay = bright;
    UPROC = set display = protect;
  FRAME-NAME = HELP.S.MENU;
    FRAME-TYPE = XEQ;
    UPROC = set padchar = $termpad;
    UPROC = set protect;
    UPROC = set tdisplay = bright;
    UPROC = set display = protect;
  FRAME-NAME = HELP.GET;
```

```
  FRAME-TYPE = XEQ;
  UPROC = set padchar = $termpad;
  UPROC = set protect;
FRAME-NAME = SHOWHELP;
  FRAME-TYPE = DATA;
  UPROC = set protect;
```

APPENDIX 7. NEWSPAPER INDEX FILE DEFINER

```
- ************************************************************************************
file NEWSPAPR:NEWSPAPERINDEX/ author Harry M. Kriz, University Libraries, 703-231-7052,
KRIZ AT VTVM1/ bin purge/ statistics 2
  com  *
  com  **********************************************************************
  com  * IMPORTANT:  The generated file definition will be edited.
  com  *
  com  * EDIT THE GENERATED FILE DEFINITION AS FOLLOWS BEFORE COMPILING:
  com  *   See comments under TITLE and SUBJECT in REC01.
  com  *   Record section:
  com  *     REC01: Under ELEM = TITLE
  com  *              Delete word SUBJECT from INDEX =
  com  *            Under ELEM = SUBJECT
  com  *              Delete word SUBJECT from INDEX =
  com  *     REC02: Delete REMOVED
  com  *              Otherwise the subject heading key is unnecessarily
  com  *              duplicated in the residual
  com  *     ZIN04: Delete ALIASES = SUBJECT
  com  *   Linkage section:
  com  *     ZIN04: (TITLE index)  Delete word SUBJECT from SEARCHTERMS
  com  *   Subfile section:
  com  *     Clean up duplicated comments
  com  **********************************************************************
  com  *
  com  *  PURPOSE:  An index to articles in the Collegiate Times,
  com  *            News Messenger, Roanoke Times, and Spectrum.
  com  *            Data maintained by Reference Department.
  com  *
  com  *            Initial data loading from data converted from
  com  *            20 ProCite databases.
  com  *
  com  BIN PURGE discards overnight processing messages unless a problem occurs.
  com  STATISTICS 2 writes logging information to a CMS file during overnight
  com  processing when subfile logging is turned on in the subfile section.
  com  ********************************************************************
  com  *       Remember to add logging to public subfile section
  com  ********************************************************************
  com  .  Designed by H. M. Kriz, R. Stelk, D. Beagle, B. Obenhaus
  com  .  Written:  8/6/91          By:  H. M. Kriz
  com  *******  END COMMENTS INCLUDED IN THE FILE DEFINER  *************
- ============================================================================
goal REC01/ result Article, Articles/ passer KEY
- ----------------------------------------------------------
subfile NEWSPAPER INDEX
  cmd set format $prompt + da
  cmd show select
  cmd show subfile size
      exp
      exp  NEWSPAPER INDEX is an index to articles about southwest
      exp  Virginia, Blacksburg, and Virginia Tech which have appeared
      exp  in the Christiansburg News Messenger, Collegiate Times,
      exp  Roanoke Times, and Virginia Tech Spectrum.
      exp
- ----------------------------------------------------------
FIXED
key ID/ slot
ele DATE.ADDED, da/ date add/ single/ index/ msg
      exp
      exp  SPIRES will supply today's date if you make no entry.
      exp
- ----------------------------------------------------------
REQUIRED
ele TITLE, t, keyword, kw/ text/ squeeze/ single/ index/ word/ exclude
```

```
      exp
      exp  Enter the title of the article exactly as you want it to
      exp  appear.  Only one title may be entered.  Each word will
      exp  be indexed in the TITLE index, also known as the KEYWORD
      exp  index.  Words from the subject headings are also included
      exp  in this index.
      exp
      com  ************************************************************
      com  Delete the word SUBJECT from INDEX = in the generated
      com  filedef for this element.  Also delete the word SUBJECT
      com  from the SEARCHTERMS = for the TITLE index (ZIN04).
      com  ************************************************************
      com
ele SOURCE, src/ squeeze/ capitalize/ single/ include CT, NM, RT, S/ index/ +
       msg Valid entries are CT, NM, RT, S
      OUTPROC $CHANGE.LIST('CT,''Collegiate Times'',NM,''News Messenger'', +
           RT,''Roanoke Times'',S,Spectrum')
      exp
      exp  Enter the code for the newspaper. SPIRES will substitute the
      exp  full name when the data is output.
      exp
      exp    CT = Collegiate Times        NM = News Messenger
      exp    RT = Roanoke Times            S = Virginia Tech Spectrum
      exp
ele DATE, d/ date/ single/ index/ msg
      exp
      exp  Enter the date the article was published
      exp  using any valid date format.
      exp
ele PAGE, p/ text/squeeze/ capitalize/ single/ msg
      exp
      exp  Enter the page where the article starts using any combination
      exp  of text and numbers.
      exp
ele SUBJECT, subj, sh/ text/ squeeze/ capitalize/ lookup verify,2/ +
           index title/ closeout $sort(ascend)/ +
           msg Only valid subjects from the NEWSPAPER SUBJECTS subfile are accepted.
      exp
      exp  Enter as many valid subject headings as desired for this article.
      exp  Only valid subject headings from the NEWSPAPER SUBJECTS subfile
      exp  are accepted. Each heading is entered in a separate occurrence
      exp  of this element. Each word in the heading will be indexed in
      exp  the TITLE index, also known as the KEYWORD index.
      exp
      com  Valid subject headings are entered in the subfile
      com  NEWSPAPER SUBJECTS which, is REC02 in this file.
      com  The record number is used instead of the record name in the
      com  lookup because this element is indexed, and the record name
      com  will not work correctly for an indexed element in a
      com  forward lookup.
- ------------------------------------------------------------
OPTIONAL
ele AUTHOR, a/ name/ single/ index
      exp
      exp  Enter the name of the author of the article, if any.
      exp  Only one name may be entered.
      exp
ele NOTES, n/ text/ squeeze/ single/ msg
      exp
      exp  Enter any explanatory notes about the article.
      exp
- ============================================================
goal REC02/ result Subject, Subjects
- ------------------------------------------------------------
subfile NEWSPAPER SUBJECTS
  cmd set format $prompt subjhead
  cmd show select
  cmd show subfile size
      exp
      exp  NEWSPAPER SUBJECTS is a list of valid subject headings used
      exp  in the SUBJECT element in the subfile NEWSPAPER INDEX.
      exp
```

```
- ------------------------------------------------------
REQUIRED
key SUBJECT.HEADING, subjhead,subhd/ text/ squeeze/ cap
     exp
     exp  Enter any phrase to be used as a subject heading in the
     exp  NEWSPAPER INDEX.
     exp
- ------------------------------------------------------
OPTIONAL
ele DUMMY/ text/ squeeze/ single
     exp
     exp  Place holder for possible future use.
     exp
- ======================================================
END
```

APPENDIX 8. NEWSPAPER INDEX FILE DEFINITION

```
FILE = NEWSPAPR:NEWSPAPERINDEX;
COMMENTS = Generated by FILE DEFINER 08/06/91;
COMMENTS = *;
COMMENTS = ************************************************************************;
COMMENTS = * IMPORTANT:  The generated file definition will be edited.;
COMMENTS = *;
COMMENTS = * EDIT THE GENERATED FILE DEFINITION AS FOLLOWS BEFORE COMPILING:;
COMMENTS = *   See comments under TITLE and SUBJECT in REC01.;
COMMENTS = *   Record section:;
COMMENTS = *     REC01: Under ELEM = TITLE;
COMMENTS = *              Delete word SUBJECT from INDEX =;
COMMENTS = *            Under ELEM = SUBJECT;
COMMENTS = *              Delete word SUBJECT from INDEX =;
COMMENTS = *     REC02: Delete REMOVED;
COMMENTS = *              Otherwise the subject heading key is unnecessarily;
COMMENTS = *              duplicated in the residual;
COMMENTS = *     ZIN04: Delete ALIASES = SUBJECT;
COMMENTS = *   Linkage section:;
COMMENTS = *     ZIN04: (TITLE index)  Delete word SUBJECT from SEARCHTERMS;
COMMENTS = *   Subfile section:;
COMMENTS = *     Clean up duplicated comments;
COMMENTS = ************************************************************************;
COMMENTS = *;
COMMENTS = *  PURPOSE:  An index to articles in the Collegiate Times,;
COMMENTS = *            News Messenger, Roanoke Times, and Spectrum.;
COMMENTS = *            Data maintained by Reference Department.;
COMMENTS = *;
COMMENTS = *            Initial data loading from data converted from;
COMMENTS = *            20 ProCite databases.;
COMMENTS = *;
COMMENTS = BIN PURGE discards overnight processing messages unless a problem occurs.;
COMMENTS = STATISTICS 2 writes logging information to a CMS file during overnight;
COMMENTS = processing when subfile logging is turned on in the subfile section.;
COMMENTS = ************************************************************************;
COMMENTS = *       Remember to add logging to public subfile section;
COMMENTS = ************************************************************;
COMMENTS = .  Designed by H. M. Kriz, R. Stelk, D. Beagle, B. Obenhaus;
COMMENTS = .  Written:  8/6/91          By:  H. M. Kriz;
COMMENTS = *******  END COMMENTS INCLUDED IN THE FILE DEFINER  *************;
COMMENTS = Record changes, other than changes to the SUBFILE section -;
COMMENTS = Modified: 9/17/91 by H. Kriz.  Added $CHANGE.LST INPROC   -;
COMMENTS = to SOURCE to allow TRA/UPD transaction.                   -;
COMMENTS = Modified:  10/23/91 by H. Kriz. Added $SEARCH.TRUNC       -;
COMMENTS = searchproc to allow wild card search on keywords.         -;
COMMENTS = Modified $WORD searchproc to allow use of * as truncation -;
COMMENTS = character.  Otherwise it would be converted to a blank.   -;
COMMENTS = -;
COMMENTS = Modified:  11/23/91 by H. Kriz.  Treat hyphen as space    -;
COMMENTS = Add BREAK.HYPHEN to SEARCHPROC and PASSPROC for ZIN04     -;
COMMENTS = Copied from FILEDEF on VM1, 3/16/92, by H. Kriz           -;
COMMENTS = ************************************************************;
AUTHOR = Harry M. Kriz,University Libraries,703-231-7052,KRIZ AT VTVM1;
DEFDATE = TUES. AUG. 6, 1991;
MODDATE = THUR. SEPT. 24, 1992;
MODTIME = 16:26:38;
BIN = PURGE;
STATISTICS = 2;
RECORD-NAME = REC01;
  REMOVED;
  SLOT;
    SLOT-NAME = ID;
    ELEMINFO;
      VALUE-TYPE = NUMERIC;
```

```
  FIXED;
    ELEM = DATE.ADDED;
      OCCURS = 1;
      LENGTH = 4;
      INPROC = $MSG(Invalid date value)/ $DATE/ $GEN.DATE(ADD);
      OUTPROC = $DATE.OUT;
      ALIASES = DA;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " SPIRES will supply today's date if you make no entry.";
        DESCRIPTION;
        SUPPLIED = Today's Date;
        VALUE-TYPE = DATE;
        INDEX = DATE.ADDED;
        INDEX = DA;
  REQUIRED;
    ELEM = TITLE;
      OCCURS = 1;
      INPROC = $SQU;
      ALIASES = T, KEYWORD, KW;
      COMMENTS = ************************************************************;
      COMMENTS = Delete the word SUBJECT from INDEX = in the generated;
      COMMENTS = filedef for this element.  Also delete the word SUBJECT;
      COMMENTS = from the SEARCHTERMS = for the TITLE index (ZIN04).;
      COMMENTS = ************************************************************;
      COMMENTS;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the title of the article exactly as you want it to";
        DESCRIPTION = " appear.  Only one title may be entered.  Each word will";
        DESCRIPTION = " be indexed in the TITLE index, also known as the KEYWORD";
        DESCRIPTION = " index.  Words from the subject headings are also included";
        DESCRIPTION = " in this index.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
        INDEX = TITLE;
        INDEX = T;
        INDEX = KEYWORD;
        INDEX = KW;
    ELEM = SOURCE;
      OCCURS = 1;
      INPROC = $CHANGE.LIST('''Collegiate Times'',CT,''News Messenger'',NM,''Roanoke
Times'',RT,Spectrum,S')/
   $MSG('Valid entries are CT,NM,RT,S')/ $SQU/ $CAP/ $INCLUDE('CT, NM, RT, S');
      OUTPROC = $CHANGE.LIST('CT,''Collegiate Times'',NM,''News Messenger'',
RT,''Roanoke
Times'',S,Spectrum');
      ALIASES = SRC;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the code for the newspaper. SPIRES will substitute the";
        DESCRIPTION = " full name when the data is output.";
        DESCRIPTION;
        DESCRIPTION = "   CT = Collegiate Times        NM = News Messenger";
        DESCRIPTION = "   RT = Roanoke Times            S = Virginia Tech Spectrum";
        DESCRIPTION;
        INDEX = SOURCE;
        INDEX = SRC;
    ELEM = DATE;
      OCCURS = 1;
      LENGTH = 4;
      INPROC = $MSG(Invalid date value)/ $DATE;
      OUTPROC = $DATE.OUT;
      ALIASES = D;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the date the article was published";
        DESCRIPTION = " using any valid date format.";
        DESCRIPTION;
        VALUE-TYPE = DATE;
        INDEX = DATE;
        INDEX = D;
```

```
    ELEM = PAGE;
      OCCURS = 1;
      INPROC = $SQU/ $CAP;
      ALIASES = P;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the page where the article starts using any combination";
        DESCRIPTION = " of text and numbers.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
    ELEM = SUBJECT;
      INPROC = $MSG('Only valid subjects from the NEWSPAPER SUBJECTS subfile are
accepted.')/ $SQU/ $CAP/
$LOOKUP(verify,2)/ $sort(ascend);
      ALIASES = SUBJ, SH;
      COMMENTS = Valid subject headings are entered in the subfile;
      COMMENTS = NEWSPAPER SUBJECTS which, is REC02 in this file.;
      COMMENTS = The record number is used instead of the record name in the;
      COMMENTS = lookup because this element is indexed, and the record name;
      COMMENTS = will not work correctly for an indexed element in a;
      COMMENTS = forward lookup.;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter as many valid subject headings as desired for this
article.";
        DESCRIPTION = " Only valid subject headings from the NEWSPAPER SUBJECTS subfile";
        DESCRIPTION = " are accepted. Each heading is entered in a separate occurrence";
        DESCRIPTION = " of this element. Each word in the heading will be indexed in";
        DESCRIPTION = " the TITLE index, also known as the KEYWORD index.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
        INDEX = TITLE;
        INDEX = T;
        INDEX = KEYWORD;
        INDEX = KW;
  OPTIONAL;
    ELEM = AUTHOR;
      OCCURS = 1;
      INPROC = $NAME;
      OUTPROC = $NAME;
      ALIASES = A;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the name of the author of the article, if any.";
        DESCRIPTION = " Only one name may be entered.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
        INDEX = AUTHOR;
        INDEX = A;
    ELEM = NOTES;
      OCCURS = 1;
      INPROC = $SQU;
      ALIASES = N;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter any explanatory notes about the article.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
RECORD-NAME = REC02;
  REQUIRED;
    KEY = SUBJECT.HEADING;
      OCCURS = 1;
      INPROC = $SQU/ $CAP;
      ALIASES = SUBJHEAD, SUBHD;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter any phrase to be used as a subject heading in the";
        DESCRIPTION = " NEWSPAPER INDEX.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
    OPTIONAL;
      ELEM = DUMMY;
```

```
      OCCURS = 1;
      INPROC = $SQU;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Place holder for possible future use.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
RECORD-NAME = ZIN03;
  REQUIRED;
    KEY = DATE.ADDED;
      INPROC = $DATE;
      OUTPROC = $DATE.OUT;
  OPTIONAL;
    ELEM = POINTER;
      LENGTH = 4;
      INPROC = $INT;
      OUTPROC = $INT.OUT;
RECORD-NAME = ZIN04;
  COMBINE = ZIN03;
  REQUIRED;
    KEY = TITLE;
  OPTIONAL;
    ELEM = POINTER;
      LENGTH = 4;
      INPROC = $INT;
      OUTPROC = $INT.OUT;
RECORD-NAME = ZIN05;
  COMBINE = ZIN03;
  REQUIRED;
    KEY = SOURCE;
  OPTIONAL;
    ELEM = POINTER;
      LENGTH = 4;
      INPROC = $INT;
      OUTPROC = $INT.OUT;
RECORD-NAME = ZIN06;
  COMBINE = ZIN03;
  REQUIRED;
    KEY = DATE;
      INPROC = $DATE;
      OUTPROC = $DATE.OUT;
  OPTIONAL;
    ELEM = POINTER;
      LENGTH = 4;
      INPROC = $INT;
      OUTPROC = $INT.OUT;
RECORD-NAME = ZIN07;
  COMBINE = ZIN03;
  REQUIRED;
    KEY = AUTHOR;
  OPTIONAL;
    ELEM = FIRST;
      TYPE = STRUCTURE;
  STRUCTURE = FIRST;
    REQUIRED;
      KEY = FN;
        INPROC = $PNAME;
        OUTPROC = $PNAME;
    OPTIONAL;
      ELEM = POINTER;
        LENGTH = 4;
        INPROC = $INT;
        OUTPROC = $INT.OUT;
GOALREC-NAME = REC01;
  PTR-ELEM = POINTER;
    EXTERNAL-NAME = ARTICLE, ARTICLES;
      GOALREC-KEY = ID;
      PASSPROC = $PASS(NUMERIC);
  INDEX-NAME = ZIN03;
    SEARCHTERMS = DATE.ADDED, DA;
      SEARCHPROC = $MSG(Invalid date value)/ $DATE(TRUNC);
      PASSPROC = $PASS.ELEM('DATE.ADDED',NUMERIC);
```

```
      INDEXINFO;
        SOURCE = DATE.ADDED;
        VALUE-TYPE = DATE;
    PTR-GROUP = POINTER;
  INDEX-NAME = ZIN04;
    SEARCHTERMS = TITLE, T, KEYWORD, KW;
      SEARCHPROC = "$WORD('~!@#$%&()_+=|\{}[]:;""<>,.?/',BREAK.HYPHEN)/
$EXCLUDE(COMMON.WORDS)/
$SEARCH.TRUNC(*,,4,S)";
      PASSPROC = $PASS.ELEM('TITLE, SUBJECT',2)/ $WORD(PASS,BREAK.HYPHEN)/
$EXCLUDE(COMMON.WORDS);
      INDEXINFO;
        SOURCE = TITLE;
        SOURCE = SUBJECT;
        VALUE-TYPE = WORD;
        TRUNCATE = *;
    PTR-GROUP = POINTER;
  INDEX-NAME = ZIN05;
    SEARCHTERMS = SOURCE, SRC;
      SEARCHPROC = $MSG('Valid entries are CT,NM,RT,S')/ $INCLUDE('CT, NM, RT, S');
      PASSPROC = $PASS.ELEM('SOURCE',1);
      INDEXINFO;
        SOURCE = SOURCE;
    PTR-GROUP = POINTER;
  INDEX-NAME = ZIN06;
    SEARCHTERMS = DATE, D;
      SEARCHPROC = $MSG(Invalid date value)/ $DATE(TRUNC);
      PASSPROC = $PASS.ELEM('DATE',NUMERIC);
      INDEXINFO;
        SOURCE = DATE;
        VALUE-TYPE = DATE;
    PTR-GROUP = POINTER;
  INDEX-NAME = ZIN07;
    SEARCHTERMS = AUTHOR, A;
      SEARCHPROC = $PNAME(TRANS);
      PASSPROC = $PASS.ELEM('AUTHOR',NAME)/ $PNAME(TRANS,SPECIAL);
      INDEXINFO;
        SOURCE = AUTHOR;
        VALUE-TYPE = NAME;
    SUB-INDEX = FIRST;
      SEARCHTERMS = NONE;
        PRIV-TAG = 1;
        PASSPROC = $PASS.OCC;
    PTR-GROUP = POINTER;
SUBFILE-NAME = NEWSPAPER INDEX;
  EXP;
  EXP = " NEWSPAPER INDEX is an index to articles about southwest";
  EXP = " Virginia, Blacksburg, and Virginia Tech which have appeared";
  EXP = " in the Christiansburg News Messenger, Collegiate Times,";
  EXP = " Roanoke Times, and Virginia Tech Spectrum.";
  EXP;
  GOAL-RECORD = REC01;
    ACCOUNTS = NEWSPAPR;
      NOSEARCH = 1;
      SELECT-COMMAND = set format $prompt + da;
      SELECT-COMMAND = show select;
      SELECT-COMMAND = show subfile size;
      SELECT-COMMAND = set xeq NEWSPAPER PROTOCOLS;
  GOAL-RECORD = REC01;
    ACCOUNTS = PUBLIC;
      SECURE-SWITCHES = 3;
      NOSEARCH = 1;
      SELECT-COMMAND = set xeq NEWSPAPER PROTOCOLS;
      SELECT-COMMAND = ..PUBLIC;
SUBFILE-NAME = NEWSPAPER SUBJECTS;
  EXP;
  EXP = " NEWSPAPER SUBJECTS is a list of valid subject headings used";
  EXP = " in the SUBJECT element in the subfile NEWSPAPER INDEX.";
  EXP;
  GOAL-RECORD = REC02;
    ACCOUNTS = NEWSPAPR;
      SELECT-COMMAND = set format $prompt subjhead;
```

```
SELECT-COMMAND = show select;
SELECT-COMMAND = show subfile size;
```

APPENDIX 9. HELP FILE DEFINER

```
file NEWSPAPR:NPHELP/author Harry M. Kriz, +
  University Libraries, 703-231-7052, KRIZ AT VTVM1/ bin purge
- ----------------------------------------------------
goal REC01/ result Help
- ----------------------------------------------------
subfile NEWSPAPER HELP
cmd set format $prompt
cmd show select
cmd show subfile size
   exp
   exp  NEWSPAPER HELP contains the text displayed on help screens
   exp  in the NEWSPAPER INDEX subfile.
   exp
- ----------------------------------------------------
subfile NEWSPAPER HELP/ accounts PUBLIC/ switches 3
   exp
   exp  NEWSPAPER HELP contains the text displayed on help screens
   exp  in the NEWSPAPER INDEX subfile.
   exp
- ----------------------------------------------------------
REQUIRED
key HELPTOPIC/ text/ cap/ single
   exp
   exp  Enter the name of the topic as it will be called by the
   exp  NEWSPAPERPURLIC protocol.
   exp
element HELPTITLE/ text/ single
   exp
   exp  Enter the title to display on the help screen
   exp
element HELPTEXT
   exp
   exp  Enter the text of the help screen
   exp
- ----------------------------------------------------------
OPTIONAL
END
```

APPENDIX 10. HELP FILE DEFINITION

```
FILE = NEWSPAPR:NPHELP;
COMMENTS = Generated by FILE DEFINER 07/22/91;
AUTHOR = Harry M. Kriz,University Libraries,703-231-7052,KRIZ AT VTVM1;
DEFDATE = MON. JULY 22, 1991;
MODDATE = WED. AUG. 28, 1991;
MODTIME = 15:28:04;
BIN = PURGE;
RECORD-NAME = REC01;
  REMOVED;
  REQUIRED;
    KEY = HELPTOPIC;
      OCCURS = 1;
      INPROC = $CAP;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the name of the topic as it will be called by the";
        DESCRIPTION = " NEWSPAPERPUBLIC protocol.";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
    ELEM = HELPTITLE;
      OCCURS = 1;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the title to display on the help screen";
        DESCRIPTION;
        VALUE-TYPE = TEXT;
    ELEM = HELPTEXT;
      ELEMINFO;
        DESCRIPTION;
        DESCRIPTION = " Enter the text of the help screen";
        DESCRIPTION;
  SUBFILE-NAME = NEWSPAPER HELP;
    EXP;
    EXP = " NEWSPAPER HELP contains the text displayed on help screens";
    EXP = " in the NEWSPAPER INDEX subfile.";
    EXP;
    EXP;
    EXP = " NEWSPAPER HELP contains the text displayed on help screens";
    EXP = " in the NEWSPAPER INDEX subfile.";
    EXP;
    GOAL-RECORD = REC01;
      ACCOUNTS = NEWSPAPR;
    GOAL-RECORD = REC01;
      ACCOUNTS = PUBLIC;
        SECURE-SWITCHES = 3;
```

APPENDIX 11. PRINT PROTOCOL

This protocol is executed when the user presses the function key to print a set of articles found in a search. The protocol is almost independent of the database in use. Database specific items include the name of the VGROUP used by the print format and the text of the messages to the user. The print protocol could be made more generic so it could apply to any database. For example, the VGROUP name could be included in a variable in the vgroup NEWSPAPR:PUBLIC. The text of the messages could be generalized.

Code for the PRINT format and vgroup are not included here because any custom format written from scratch or generated by the GENERATE FORMAT command in $REPORT could be used with this protocol. Variables used in this protocol which are defined in the PRINT format include SaveIndex and FormatSave. Their purpose is obvious from the context of the code.

```
* PRINT (Add 09/26/91, Upd 09/24/92 at 22:21 by NEWSPAPR)
!set noecho
- ************************************************************************
-  SEE DOCUMENTATION AT END OF FILE
- ************************************************************************
allocate NEWSPAPR:PRINT       ; variables needed before format is set

if #ResultKeyCount = 0 then beginblock
                        *
                        *    There are no articles to print
                        *
                        ask prompt = '(Press <RETURN> to continue)'  NULL = '-'   attn='-'
                        endblock

else beginblock
     *
    /*  There are #ResultKeyCount titles to be printed
     *  for the search
     *
    /*     #SrchArgOK
     *
     *   Printout will be generated on the 3800 printer
     *   in the Computing Center.
     *
     REPEAT
        ask upper prompt 'Do you wish to print the list of articles (Y/N)?'
        let response = $pmatch($ask,Y?ES,N?O)
        if #response = 0 then beginblock
                              *
                              * Y or N please
                              *
                              endblock
     UNTIL #response ~= 0
     let answer = $leftstr($ask,1)

     if #answer = 'N' then beginblock       ;  User cancels printout
                     *
                    /*   Print canceled by user
                     *
                     ask prompt = '(Press <RETURN> to continue)' NULL = '-' attn='-'
                     endblock
     else beginblock                        ;  Otherwise, do the print
                          ;  If no stack due to a subsequent failed
                          ;  search and printing previously displayed
```

```
                        ;  result, then rebuild stack from Resultkey
                        ;  array
          if ~$stack then beginblock
                     let SaveIndex = #Index
                     let Index = 0
                     WHILE #Index < #ResultKeyCount
                           let Index = #Index + 1
                          /stack #ResultKey::index
                     ENDWHILE
                     let Index = #SaveIndex
                     endblock

          let FormatSave = $SETFORMAT
          set format PRINT
         /let PrintTitle = '#SrchArgOK'
     -    SET ACTIVE TEMPPRNT FILE              ;  For debugging
          setp 3800p6 pre
          set active printer (cc
          in active type
          setp restore
          SET ACTIVE ACTIVE FILE
         /set format #FormatSave
          *
         /*   #ResultKeyCount titles were printed for the search
         /*   #SrchArgOK
          *
          ask prompt = '(Press <RETURN> to contine)'  NULL = '-'  attn='-'
     endblock
endblock

deallocate NEWSPAPR:PRINT   ; variables not needed for search and display

RETURN
- ================================================================
- PRINT protocol produces a printout from the NEWSPAPER INDEX
- Written by H. M. Kriz, University Libraries, 231-7052, KRIZ @ VTVM1
- Called by print routine in PUBLIC protocol
- Calls:
-        FORMATS:  NEWSPAPR:PRINT
-        VGROUPS:  NEWSPAPR:PRINT
- Variables specific to this protocol are compiled in PRINT
- PUBLIC variables are also used.
- Modified 9/21-24/92 by H. Kriz when adapting for more generic names
- ================================================================
```

APPENDIX 12. ERRORS PROTOCOL

```
* ERRORS (Add 09/15/92, Upd 09/21/92 at 11:33 by NEWSPAPR)
!set noecho
- ************************************************************************
- SEE DOCUMENTATION AND USAGE NOTES AT END OF FILE
- ************************************************************************

if #ErrMsgNum = 212 then beginblock
     *
     *  Your keyword syntax is interpreted as
     *
    /*      #SrchArg
     *
     *  which is a form NOT understood by the system.
     *  Please edit your keywords and try again.
     endblock
else if #ErrMsgNum = 14 and #ErrSNum = 274 then beginblock
     *
     *  We're sorry, but your search request failed due to
     *  insufficient virtual storage.  This can result if
     *  you specified a very large range of dates, or
     *  otherwise issued a request which involved a
     *  Boolean AND operation involving tens of thousands
     *  of records.
     *
     *  Please edit your search request to produce a smaller
     *  search result, perhaps by specifying a more restrictive
     *  range of dates, and try again.
     *
     *  If this is not feasible, please call the Reference
     *  Desk in Newman Library at 231-6045 and ask that
     *  a consultant contact you.
     endblock
else if #ErrENum = 14 then beginblock
     *
     *  Your search cannot be executed because a
     *  truncated keyword has a stem shorter than
     *  the required 5 characters.  Please edit your
     *  keywords and try again.
     endblock
else beginblock
     *
     *  We're sorry, but an unanticipated SPIRES error
     *  occurred during your search.  You should be able to
     *  continue searching by editing your request and trying
     *  again.
     *
     *  If you would like more information and assistance with
     *  the search that caused the error, please call the
     *  Reference Department in Newman Library at 231-6045
     *  and report that you encountered SPIRES error code:
     *
    /*    Message Number:  #ErrMsgNum
    /*      Error Number:  #ErrSNum
     *
     *  while searching the $SELECT database.
     endblock
*
*  When you press the CLEAR key, you will be returned
*  to the search screen and you will see the message
*
*       'No articles found...'
*
*  You may continue searching after modifying your request.

RETURN
- ========== DOCUMENTATION AND USAGE NOTES BEGIN ==========
```

```
- ERRORS protocol displays error messages for NEWSPAPER INDEX
- Written by :  H. M. Kriz, University Libraries, 231-7052, KRIZ@VTVM1
- Called by: PUBLIC PROTOCOL for NEWSPAPER INDEX
-            Application tests an error condition. If it exists, the
-            SPIRES system variables $snum, $enum, $msgnum are copied
-            to global variables in the application and this protocol
-            is executed to display an appropriate message.
-            Values for the message numbers are listed in the SPIRES
-            system subfile SYSTEM MESSAGES
- Variables: ErrENum, ErrMsgNum, ErrSNum set by calling protocol are
-            compiled in PUBLIC vgroup used by NEWSPAPER INDEX.
-            SrchArg - the search argument
- MODIFIED: 9/21/92 by H. Kriz to use more generic names
- ========== DOCUMENTATION AND USAGE NOTES END ==========
```